THE UNIVERSITY OF ALBERTA

RELEASE FORM

NAME OF AUTHOR	Charles W. Huneycutt
TITLE OF THESIS	An Interactive Biochemical
	Simulation System
DEGREE FOR WHICH THESIS WAS PRESENTED	M.Sc.
YEAR THIS DEGREE GRANTED	1976

AN INTERACTIVE BIOCHEMICAL SIMULATION SYSTEM

by

© CHARLES W. HUNEYCUTT

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES AND RESEARCH

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE

OF MASTER OF SCIENCE

DEPARTMENT OF COMPUTING SCIENCE

EDMONTON, ALBERTA

SPRING, 1976

FACULTY OF GRADUATE STUDIES AND RESEARCH

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research, for acceptance, a thesis entitled AN INTERACTIVE BIOCHEMICAL SIMULATION SYSTEM submitted by Charles W. Huneycutt in partial fulfilment of the requirements for the degree of Master of science.

ABSTRACT

Computers can contribute much to biological research. To fully tap the potential of computer applications in the biological sciences, special purpose systems are necessary, systems that provide useful research aids and are easy to operate. Several such systems are reviewed.

An interactive system for biochemical simulation that permits dynamic modification of the model is proposed. Its structure and function are described and a partial implementation presented. A model of the lac operon of the bacterium _Escherichia_ _coli_ is described and its validation and uses are discussed.

ACKNOWLEDGEMENTS

I thank Jeff Sampson for his patience, academic guidence, and untiring assistance in preparation of this thesis. I also thank the Department of Computing Science at the University of Alberta and the National Research Council, who made the whole thing possible.

TABLE OF CONTENTS

LIST OF TABLES AND FIGURES

1. INTRODUCTION

1.1 Purpose

The use of computers for research and study in the biological sciences poses a problem. Few researchers in this field possess, or desire to acquire, the special skills necessary to design and implement large computer programs. Interactive languages like APL are gaining popularity for simple data analysis and calculations, but do not tap the full capabilities of computer applications in the biological sciences. To achieve this goal, special systems are necessary that provide powerful computation tools while requiring a minimal understanding of the actual mechanics of computation, data storage, and data structure design.

This thesis will explore the area of biochemical simulation systems, which produce simulations for a model represented as a reacting mixture of substances which combine and/or separate in proportion to the product of their concentrations, yielding other substances in the system.

1.2 Notation

The notation generally used to describe biochemical simulation systems is chemical flux equations. These equations list (1) the reacting substances; (2) their stoichiometrics, or relative proportion of combination; (3) rate constants which give the kinetic velocity of the reaction; and (4) substances which are produced by the reaction. The notation is as follows:

$$A + B \xrightarrow{K1} C$$

This equation states that one unit of A combines with two units of B to produce one unit of C. The rate of the reaction is given by K such that

$$d[C]/dt = -d[A]/dt = -d[B]/dt = K1[A][B]$$

The reverse reaction also may occur where:

$$A + B \xleftarrow{K2} C$$

and the overall reaction rate would be given by

$$d[C]/dt = -d[A]/dt = -d[B]/dt = K1[A][B] - K2[C]$$

Both reactions are usually denoted by

$$A + B \overset{K1}{\underset{K2}{\longleftrightarrow}} C$$

<u>or</u>

$$A + B \overset{K3}{\longleftrightarrow} C \quad \text{where } K3 = K1/K2$$

In biochemical systems enzymes are present which modify the rate of reaction of the substances in the system. The enzymes do this by forming compounds with the reacting substances, or substrate, thus bringing the substrate together in the proper fashion so that the reactions may occur. For a slightly modified form of the previous example, and an enzyme E, the flux equations would become

$$A + E \overset{K1}{\underset{K2}{\longleftrightarrow}} AE$$

$$AE + B \overset{K3}{\underset{K4}{\longleftrightarrow}} AEB \overset{K5}{\underset{K6}{\longleftrightarrow}} E + C$$

yielding the differential equations

$$d[A]/dt = -K1[A][E] + K2[AE]$$

$$d[AE]/dt = - d[A]/dt + K5[AEB]$$

d[B]/dt = K6[AEB] - K5[AE][B]

d[AEB]/dt = - d[B]/dt - d[C]/dt

d[C]/dt = K7[AEB] - K8[E][C]

d[E]/dt = d[A]/dt + d[C]/dt

It can be easily seen that the number and complexity of the differential equations increases rapidly as the number of substances and flux equations increase.

An example of a model that could be studied using a computer simulation system might be a simple single enzyme system using allosteric (shape changing) inhibition for control of the reaction. For the previous example, the flux equations would become:

1) $A + E \underset{K2}{\overset{K1}{\longleftrightarrow}} AE$

2) $AE + B \underset{K4}{\overset{K3}{\longleftrightarrow}} AEB \underset{K6}{\overset{K5}{\longleftrightarrow}} E + C \underset{K8}{\overset{K7}{\longleftrightarrow}} E'C$

3) $A + E' \underset{K10}{\overset{K9}{\longleftrightarrow}} AE'$

4) $AE' + B \underset{K12}{\overset{K11}{\longleftrightarrow}} AE'B \underset{K14}{\overset{K13}{\longleftrightarrow}} E' + C$

In equation 2, the enzyme E can bond with the product C to form a second form of the enzyme E'. This second form of the enzyme will also catalyze the reaction of A and B to form C, but the overall rate for formation of AE', AE'B, and the product C can be different than the other form of the enzyme E. If the overall rate is less, as the concentration of C rises, the ratio

of the concentrations of E/E' will grow smaller. This will decrease the rate of production of compound C, providing negative feedback to the system based on the concentration of product C.

This is a relatively simple example of one type of biochemical control mechanism. In a biochemical system of interest there may be several control mechanisms operating, each more complicated that the above example. To analyze the system by analytic means is likely be impractical, or impossible. To reduce the system to its representations as differential equations and program it in a common higher level computer language such as FORTRAN, ALGOL, or APL involves a considerable amount of work. Relatively small structural changes to the system could result in significant changes to differential equations, with many hours spent in re-programming.

With computer simulation systems that allow the user to describe the system as chemical flux equations, it is possible to describe and explore the behavior of systems large enough to be of interest. This thesis will discuss three existing systems, and propose a fourth which overcomes some of the shortcomings of the others.

1.3 Preview

Chapter 2 will propose three criteria that will be used in discussion of the simulation systems presented in this paper. A method for classificaton of these systems will be presented and examined.

In Chapter 3, three existing biochemical simulation systems

will be studied. Why and how each system was written will be examined. Each system will be viewed in terms of the language structure, or as the user sees the system; and in terms of the total system structure, or as the programmer sees the system. Each system will be evaluated by the three criteria, classified, and examined for strong or weak points.

A proposed, and partially implemented, simulation system will be presented in the fourth chapter. It will be classified, discussed, and evaluated in the same manner as the preceding three simulation systems, but in more detail.

In Chapter 5 a model for transcriptional control will be discussed. This model will be based on the lac operon in the bacterium _Escherichia coli_. An implementation of the model using the simulation system presented in Chapter 4 will be discussed. Comments will be made on validation of the model and possible uses of the model.

Chapter 6 will summarize the material presented in this thesis, and attempt to draw some conclusions from the presentations. Possible extensions to the system proposed in Chapter 4 will be presented and examined.

2. CRITERIA AND CLASSIFICATION

2.1 Criteria

In order to provide a framework for analysis of the simulation systems presented in this paper, the following three criteria will be used:

1) The system must meet accepted requirements for numerical accuracy of the simulation. These are currently around 1.0% [Chance, Chance, and Sellers 1960].

2) The system should have an input language that accurately describes the model. Both the input language and form of output should be intuitive and natural for the user familiar with biochemical notation.

3) The system should be readily available and easy to use.

The first criterion, numerical accuracy, is necessary to insures that there will be a valid correspondence between the simulation of the model and the true properties of the model. Since the validity of a biochemical model is often based on observed data with greater than 1% degree of error, accuracy beyond 1% is not necessary.

The first part of the second criterion, accuracy of description, requires that the input language of the simulation system be able to correctly describe the esential properties of the real system being modeled. If the input language cannot do this, any correspondence between the simulation of the model and

[illegible]

the real system being modeled is lost.

The second part of the second criterion, "intuitiveness and naturalness" of description, is meant to measure the ease of describing the model and interpreting the simulation results. This is no less important than accuracy of description, for a cumbersome input language will result in errors in the model description, and awkward forms of output will make it difficult to judge the validity of the model. A natural and intuitive input language should let the user describe the model in terms with which he is familiar. It should convey the structure and function of the system clearly, and not require extensive recoding from commonly accepted notations for the class of models being studied. The output should be clear and concise and allow the user to selectively view different aspects of the model. Graphical output seems to be the most widely accepted.

The last criterion, ease and availability of use, is meant to judge the practicality of each system. If a simulation system is difficult to use, regardless of its good features, it will not be popular. This problem is much too common in computer applications. The amount of technical information that must be learned to operate a simulation system should be kept to a minimum. This frees the user to concentrate on the model and interpretation of simulation data.

For a simulation system to be useful, it must be available for use. Special hardware requirements, or restriction to a particular computer reduce the chances of being able to implement the system. If the simulation system cannot be implemented, it certainly will not be used.

The more readily accessible the simulation system is, the more useful it becomes. In this respect, interactive systems are without equal. They allow the user to construct and test the model directly on the computer, reducing the amount of time required to get usable simulation results. The user can quickly test alternate models, rejecting those that prove unsatisfactory. In this fashion, the model can be developed dynamically, based on its performance in differing situations.

2.2 Classification of Systems

Several simulation systems for biochemical models have been written. These systems have generally become more powerful and sophisticated as computers have become larger and more widely used. This thesis will classify biochemical simulation systems into three general categories:

1) Analogue and hybrid systems, where actual solutions to differential equations representing models are obtained on an analogue computer.
2) Digital-analogue simulations, where actual solutions to differential equations representing models are obtained on a digital computer programmed to mimic an analogue computer.
3) Digital simulations, where the simulation of the model is performed on a digital computer, with no attempt to make the digital computer resemble an analogue computer.

Analogue and hybrid computers naturally lend themselves to solution of differential equations. These were some of the

first computers used in the simulation of biochemical models. They can produce solutions to sets of differential equations within accuracy limitations of 0.1 to 15%, depending upon the characteristics of the individual elements of the analogue circuitry. Simulations can be run in real time (or faster) through the use of parallel circuits that represent the differential equations describing the model. Unfortunately, analogue machines are cumbersome to program, and much less available than digital computers. The development of digital-analogue hybrids has increased the usefulness of analogue simulation. However, only very large simulations on analogue or hybrid computers retain a speed and/or cost advantage over modern digital computer simulations. With the advent of micro-processors and multi processor digital computers, any remaining speed or cost advantage associated with the use of analogue equiptment is rapidly disappearing.

As digital computers became more sophisticated and more widely used, several systems were developed that mimicked the analogue computer primitives for solution to differential equations. They eliminated the cumbersome patchboards used by analogue computers, replacing them with languages that described the various analogue primitives and their inter-relationships, such as simulation and integration. Translators were written that could take a higher level language for input, and output the proper translation in analogue primitives, which in turn could be used by the analogue simulation program on the digital computer. While this arrangement seems cumbersome, it produced impressive results.

Digital simulation systems differ from the digital-analogue systems in that they do not translate the differential equations representing the model into analogue primitives. The differential equations are constructed in algebraic form, and solved by a step-wise iterative method. Digital systems allow the user to describe the biochemical system being studied in a familiar form that is closer to generally accepted notation, translating this form to the differential equations for simulation. This produces a more direct program flow, which is highly desirable for interactive systems. Reducing the number of translations necessary before the actual simulation occurs makes it possible to structurally modify the model during simulation. This permits a wider range of control mechanisms to be modeled, increasing the simulation system's usefulness.

The three criteria presented in this chapter will be applied to each system studied in this thesis. Particular emphasis will be placed on intuitiveness and naturalness of the input/output structure for each system, and its ease and availability of use. These factors are of primary importance for any computer system designed for specific applications. If they are not met, regardless of other attributes of the system, the system will not be readily accepted.

3. PREVIOUS SYSTEMS

3.1 The Johnson Foundation Electric Analog Computer, Mark II

3.1.1 Introduction. The Johnson Foundation Electronic Analog Computer, Mark II, is a hybrid computer designed specifically for simulation of chemical equations, and is used at the Johnson Foundation for the study of biochemical systems. It is intended to combine the speed of analogue simulation with the logical capabilities of digital simulation. Simulations can be run with less than 1% error. The Mark II produces rapid solutions for for sets of chemical flux equations. For example, the solution of a system of 25 reactants required only 0.1 seconds of compute time on the Mark II. A simplified patch board has been devised that allows formulation of the model as chemical equations for the analogue section [Higgens 1965].

3.1.2 Language Description. The model is formulated as a set of chemical equations of the form:

$$A + B \dashrightarrow C + D$$

These equations are then patched into the programming board, partially shown in Figure 3.1. Each horizontal row of plugs represents an equation of the above form. The first plug on the left represents A; the second on the left, B; the first on the right, C; and last on the right, D. The sockets to the side of the plugs represent the reactant concentrations. The equation is formed by inserting the appropriate plug into the desired reactant socket. Figure 3.1 shows the patching for the equation A + B ---> C where A is reactant 1, B is 2, and C is 3.

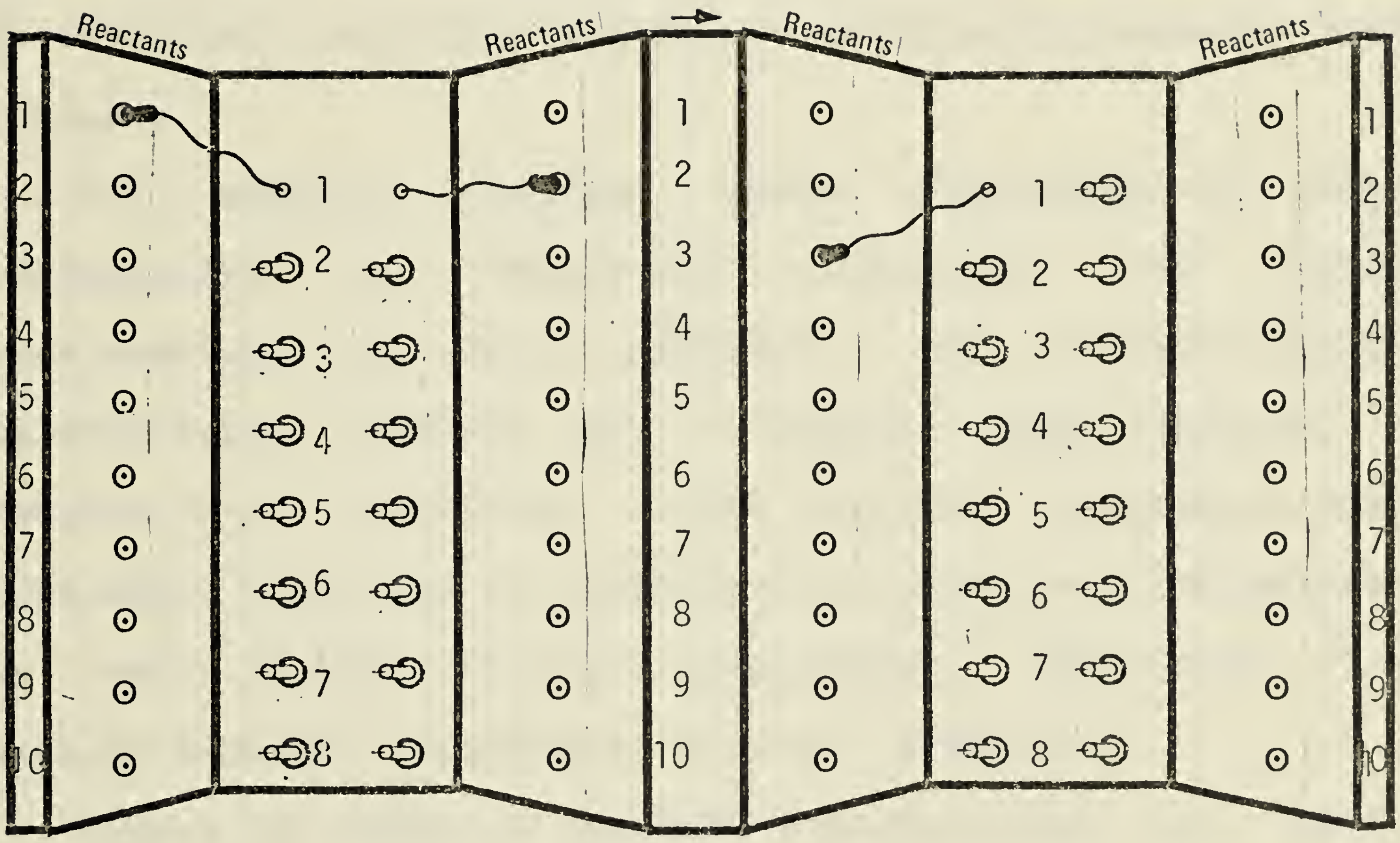


Figure 3-1 Mark II patchboard

This method is an improvement over the patch boards of general purpose analogue computers, but still requires repatching for each new model or structural changes to existing models.

Reactant concentrations and rate constants are set manually by potentiometers on the computer console. These are accurate to three decimal digits and may vary over a range of one million. It is possible to alter initial reactant concentrations and rate constants automatically through control programs.

The control programs allow monitoring of reactant concentration and conditional alteration of reactant concentrations and rate constants. This is done through an analogue curve analyzer and a digital logic section. The programs are constructed on the automatic programming control plugboard. There are no facilities to alter the structure of the model through control programming. This must be done manually through the chemical reaction plugboard.

Output is graphs of reactant concentrations vs. time on CRT monitors. Each reactant's concentration is shown on a separate CRT monitor, with a special high precision CRT monitor that can display up to eight reactant concentration graphs simultaneously. A synchronized camera is provided for hard copy output. There is no facility for direct readout of reactant concentrations. Other than photographically, there are no facilities for storage of simulation results.

To simulate a model, the user first describes the model in chemical equations of the proper form. These are transferred to the chemical reaction plugboard. The initial concentrations and

rate constants are set, and any analytic alteration functions specified at the automatic programming control plugboard. The computer is then started and the user examines the simulation results on the CRT monitors. By manual or automatic control the behavior of the model may be examined over a large range of conditions. Model parameters, such as rate constants and initial reactant concentrations, may be adjusted to produce a desired pattern of behavior, and models can be verified for set conditions.

3.1.3 <u>System Structure</u>. The Mark II is a hybrid system using analogue circuits for simulation of the model, and a small digital computer with analogue-to-digital channels for logic functions. It can be divided into the logical segments shown in Figure 3.2.

Input to the analogue section of the Mark II consists of the model's structural description from the chemical reaction patchboard, and of values for rate constants and initial reactant concentrations from the console potentiometers. The analogue model description is built from standard integration, summation and multiplication circuits. The system has a maximum capacity of 14 bimolecular or 6 monomolecular reactions and 25 reactants. This ai adequate for small simulations, but cannot compete with the capacity of modern digital computer simulation systems. The integrators, adders and multipliers in the analogue section are accurate within .1 to 1%. The time for a simulation can vary between 10 milliseconds and 32 seconds. The upper limit represents maximum allowable drift in the analogue circuitry. All equation computations are carried out in

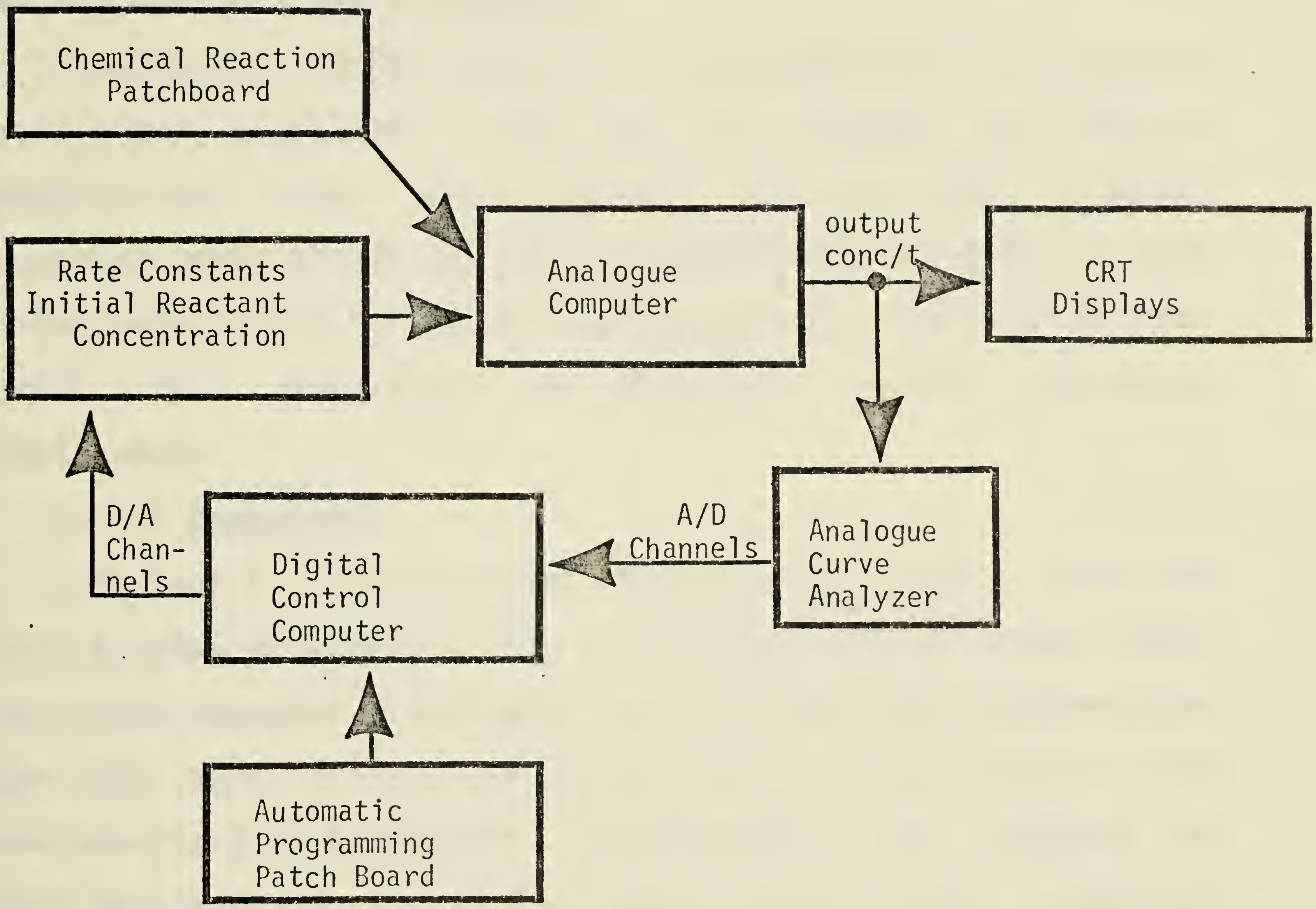


Figure 3-2 Mark II logic structure

parallel, producing excellent response times.

The analogue curve analyzer monitors the output of the analogue section and analyzes the reactant concentrations in respect to user-specified analogue functions. The output of the analysis section is fed to the digital control section through the analogue to digital channels.

The digital control section is programmed via the automatic programming patchboard. The user can program the digital section to alter rate constants and to alter reactant concentrations through the digital-to-analogue channels, based on the inputs from the curve analysis section. This can be done during the simulation, or repeatedly between successive simulations.

3.1.4 Evaluation.

The Mark II can accurately describe the model, but the input patchboard format is not an intuitive form of input, which is a major drawback in the Mark II. Arranging the patchboard to ease the task of transcribing the model from the kinetic flux equations is very innovative. Unfortunately, the plugging of cords into sockets, and setting potentiometers is not a familiar method for model description.

The Mark II is a one-of-a-kind computer. To duplicate the simulation of a model on the Mark II would require another Mark II or reprogramming for another analogue or hybrid system. Any advantages in speed of simulation in the Mark II are rapidly dissapearing with the advent of simulation systems for large digital computers.

3.2 BIOMOD

3.2.1 Introduction. BIOMOD is an interactive system for the simulation of biochemical models on a digital computer. The system will be classified as a digital-analogue system since biochemical models are translated to analogue type primitives for translation by the IBM 360 Continuous Simulation Modeling Program (CSMP) [Clark and Groner 1971; IBM 1968].

Work was begun on simulation of chemical systems at the Rand Corporation in 1958. An interactive system which could translate chemical equations into a proper set of differential equations was developed in 1965. This system led to the development of BIOMOD by G.F. Groner, R.L. Clark and R.A. Berman in 1971 [Deland 1971].

The system was designed to make full use of available software to form a user-oriented system for biological modeling. To do this, Rand's Grail system for man-machine communication was mated to IBM's CSMP program, and the FORTRAN Compiler. The result is a fully interactive system that allows the user to develop biological models as flow diagrams. Each section of these flow diagrams may be specified by chemical equations, differential equations, or FORTRAN routines.

The user can monitor the simulation of the model, and change model parameters during simulation. There are facilities for storage and retrieval of models on secondary storage devices. Output is tabular and graphical on the interactive devices' display, with hard copy available on request.

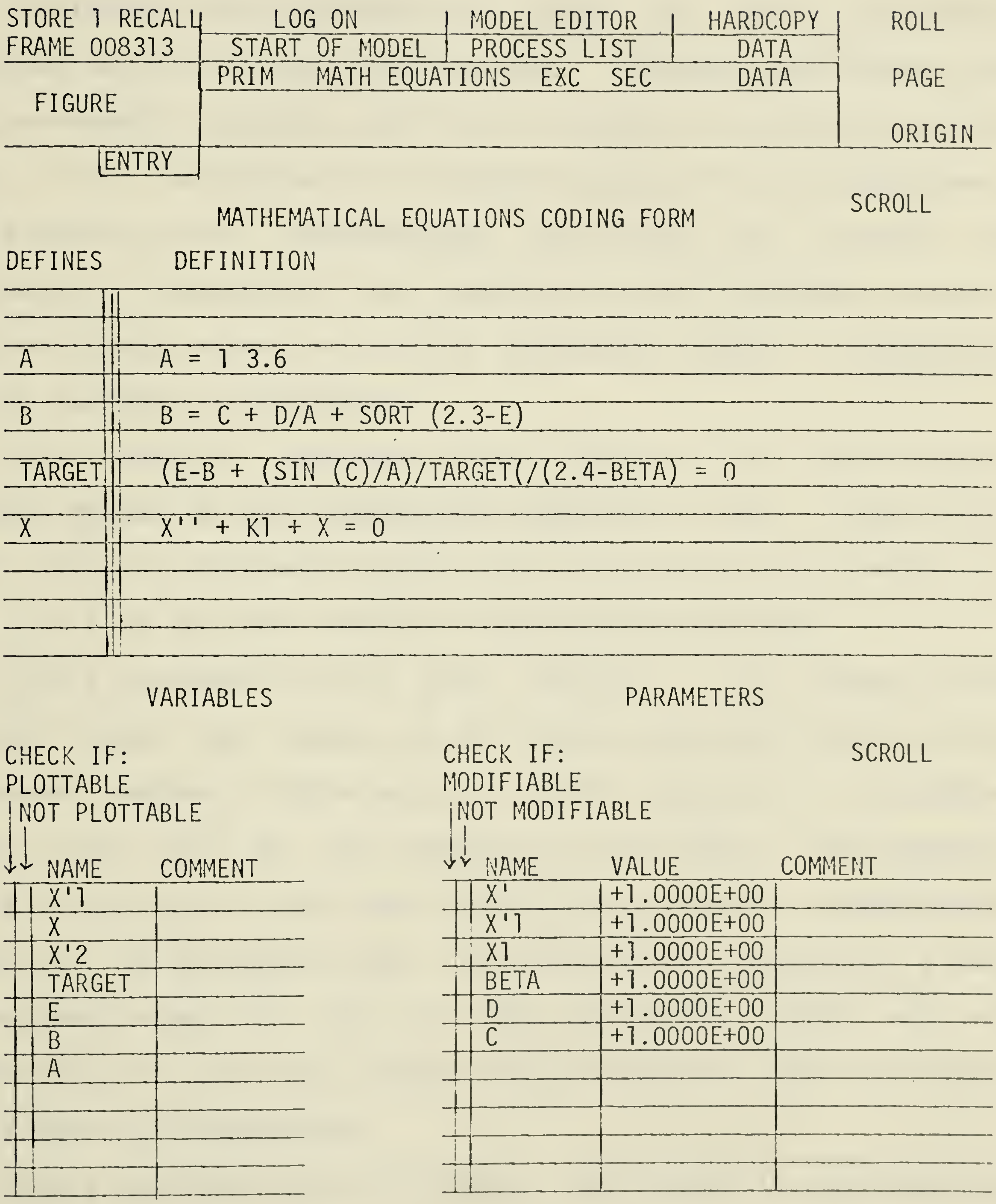
The system is fairly machine dependent because of the sophisticated nature of its interactive format. Simulations can

be run with less than 1% error [Clark, Groner, and Berman 1971], which meets current requirements.

3.2.2 <u>Language Description</u>. A model is described as a flow diagram composed of boxes connected by flow lines. Each box represents a process in the model. The flow lines lines represent the flow of materials from one model process to another. An oval on a flow line specifies which materials move between processes.

The user would first construct the model by drawing boxes and flow lines on a Rand Tablet, which is a device for free hand drawing input. The boxes may be named or unnamed. When named, the box represents a defined process which may be used as a primitive in other boxes. There are pre-defined processes representing CSMP functions and user-defined processes. Once the system structure is defined, the user completes the model by describing each user-defined process on one of the three coding forms. These forms allow the description of each process as chemical equations, algebraic or differential equations, or FORTRAN statements. Each statement or equation is scanned for syntactic correctness when entered.

The mathematical equations form (Figure 3.3) is used to enter algebraic or differential equations. As with the other two forms, it is divided into three sections: the statement section, variable table, and parameter table. The statement section contains each defined variable under 'DEFINES,' and its equation under 'DEFINITION.' The equation may be either algebraic or differential, expressed in a FORTRAN-like format. Prime signs following variables indicate derivatives with



STORE 1 RECALL FRAME 008313	LOG ON	MODEL EDITOR	HARDCOPY	ROLL
	START OF MODEL	PROCESS LIST	DATA	
FIGURE	PRIM MATH EQUATIONS EXC SEC		DATA	PAGE
ENTRY				ORIGIN

SCROLL

MATHEMATICAL EQUATIONS CODING FORM

DEFINES	DEFINITION
A	A = 1 3.6
B	B = C + D/A + SORT (2.3-E)
TARGET	(E-B + (SIN (C)/A)/TARGET(/(2.4-BETA) = 0
X	X'' + K1 + X = 0

VARIABLES

CHECK IF:
PLOTTABLE
NOT PLOTTABLE

		NAME	COMMENT
		X'1	
		X	
		X'2	
		TARGET	
		E	
		B	
		A	

PARAMETERS

CHECK IF:
MODIFIABLE
NOT MODIFIABLE

SCROLL

		NAME	VALUE	COMMENT
		X'	+1.0000E+00	
		X'1	+1.0000E+00	
		X1	+1.0000E+00	
		BETA	+1.0000E+00	
		D	+1.0000E+00	
		C	+1.0000E+00	

Figure 3-3 BIOMOD mathematical equations form

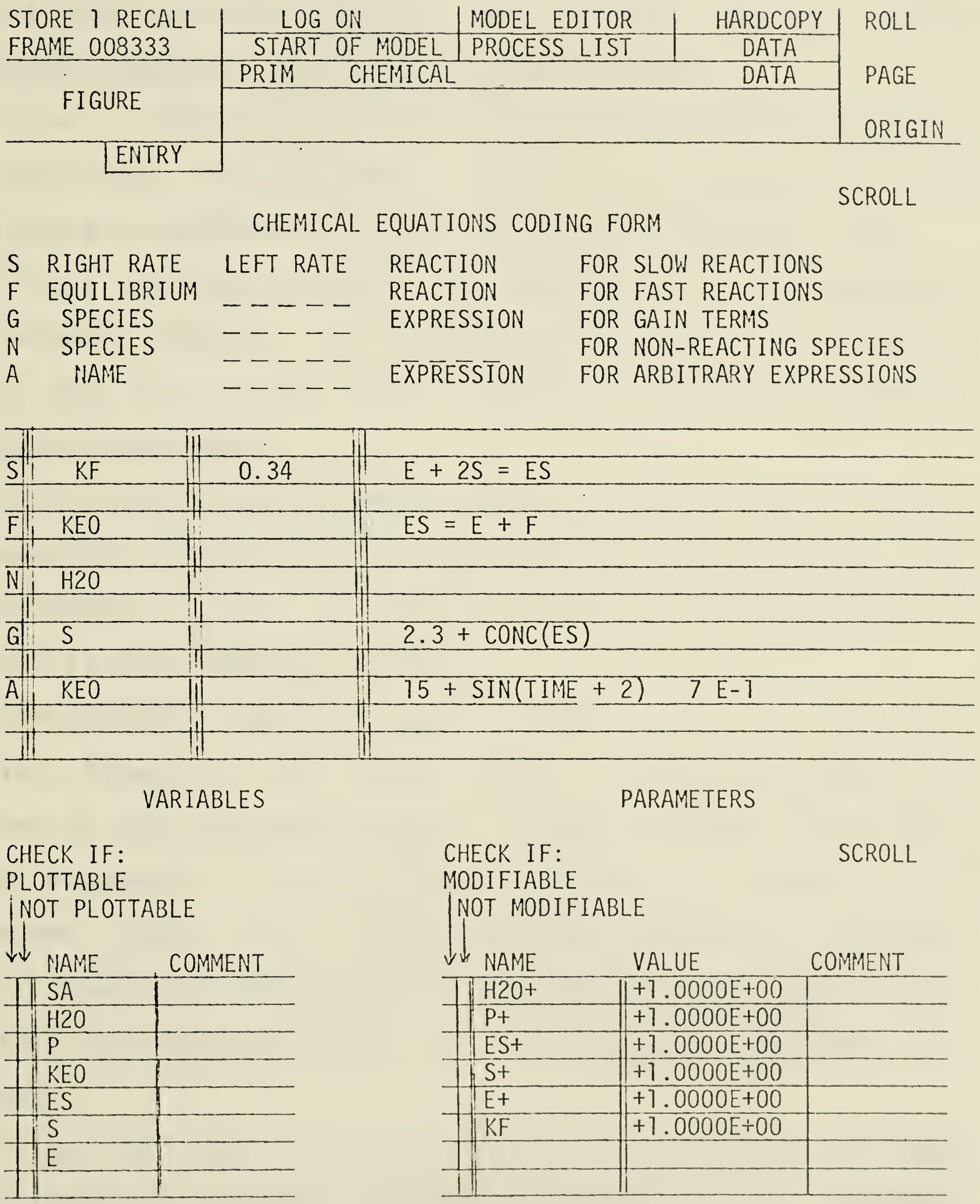
respect to time. The highest derivative of the defined variable may appear only once, and may not be the argument of functions such as square root or integration.

Variables and parameters are listed in their respective sections as the equations are entered. Variables are defined in the statements' contexts, and may be flagged as plottable by the user. If a derivative of a variable appears in its definition, the highest and all intermediate derivitives are treated as variables. Parameters are undefined in the statement context and have their values set in the parameter table. Parameters may be flagged as modifiable.

The chemical equations form (Figure 3.4) has the same general format as the mathematical equations form. There are five types of statements which are indicated by the letters, S, F, G, N, A in the left field of the statement section.

The S statement is for slow reaction. The forward and reverse rates are given in the second and third field of the statement section. These may be a numeric value or a variable whose value may be set elsewhere in the model. The chemical equation is given in the last field as two chemical expressions separated by an equals sign. Each expression consists of a sum of reactant names of one to four characters which may be preceded by a two digit integer stoichiometric. Slow reactions are solved by integration.

The F indicates fast reactions. The format is identical to slow reactions except that an equilibrium constant is given in the second field rather than rate constants. Fast reactions are forced to equilibrium at each time step by matrix operations.



STORE 1 RECALL FRAME 008333	LOG ON / START OF MODEL	MODEL EDITOR / PROCESS LIST	HARDCOPY / DATA	ROLL
FIGURE	PRIM CHEMICAL		DATA	PAGE
ENTRY				ORIGIN

SCROLL

CHEMICAL EQUATIONS CODING FORM

S	RIGHT RATE	LEFT RATE	REACTION	FOR SLOW REACTIONS
F	EQUILIBRIUM	_ _ _ _ _ _	REACTION	FOR FAST REACTIONS
G	SPECIES	_ _ _ _ _ _	EXPRESSION	FOR GAIN TERMS
N	SPECIES	_ _ _ _ _ _	_ _ _ _	FOR NON-REACTING SPECIES
A	NAME	_ _ _ _ _ _	EXPRESSION	FOR ARBITRARY EXPRESSIONS

S	KF	0.34	E + 2S = ES
F	KEO		ES = E + F
N	H2O		
G	S		2.3 + CONC(ES)
A	KEO		15 + SIN(TIME + 2) 7 E-1

VARIABLES

CHECK IF:
PLOTTABLE
NOT PLOTTABLE

		NAME	COMMENT
		SA	
		H2O	
		P	
		KEO	
		ES	
		S	
		E	

PARAMETERS

CHECK IF:
MODIFIABLE
NOT MODIFIABLE

SCROLL

		NAME	VALUE	COMMENT
		H2O+	+1.0000E+00	
		P+	+1.0000E+00	
		ES+	+1.0000E+00	
		S+	+1.0000E+00	
		E+	+1.0000E+00	
		KF	+1.0000E+00	

Figure 3-4 BIOMOD chemical equations form

The N indicates non-reacting species which affect the relative concentration of reactants in the process but are not involved in the reactions. The reactant name is given in the second field, and its initial value may be entered in the parameter section of the chemical equation form.

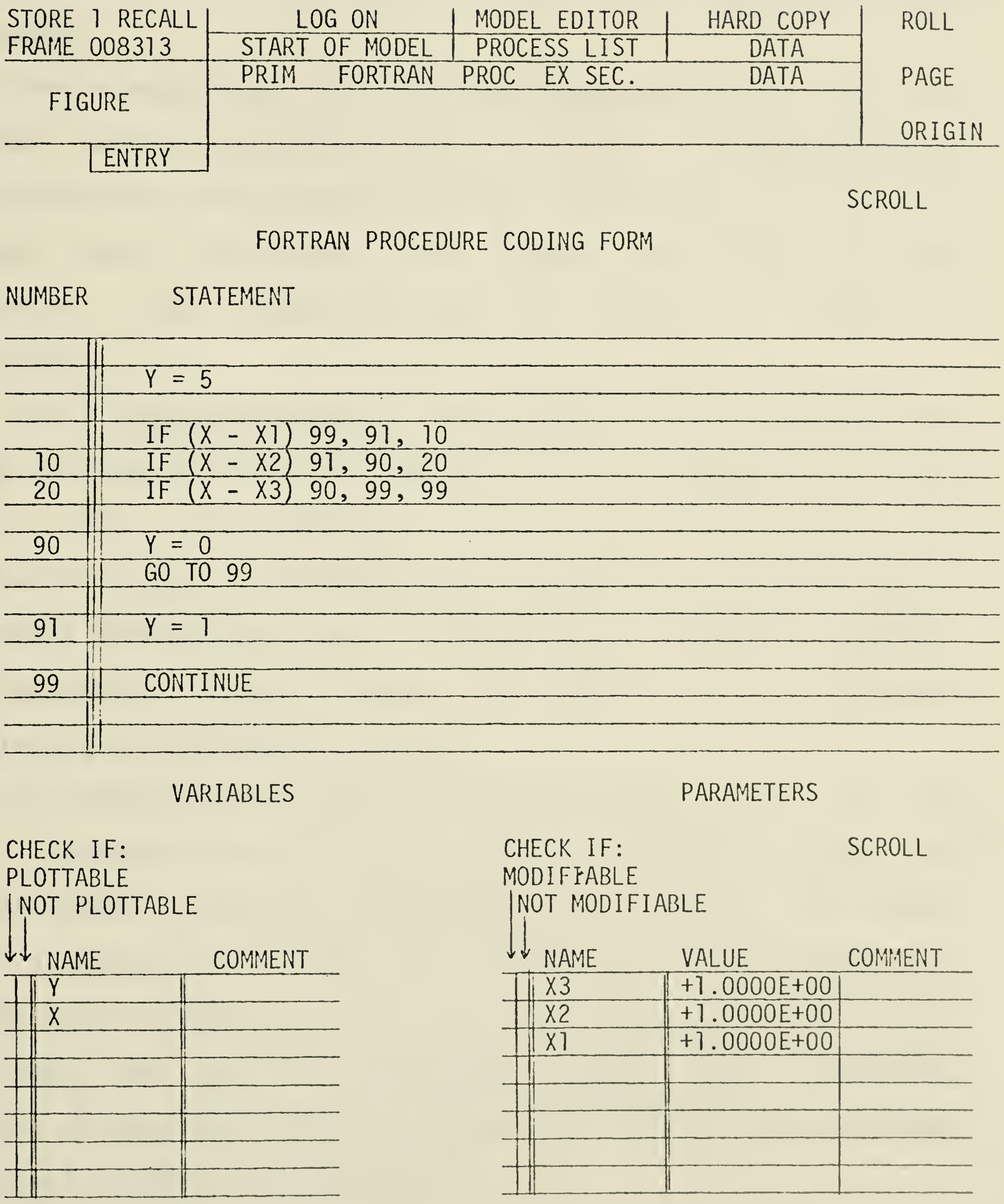
The G indicates gain for reactants which flow into or out of the process. The reactant is given in the second field, and an algebraic expression for the net flow in the fourth field.

The A indicates arbitrary expressions for the determination of variable values. The variable is specified in the second field, and the fourth field contains a FORTRAN expression specifying its value.

The variable and parameter sections are the same as in the mathematical expression form. The reactants and parameters are automatically posted to these sections as the statements are entered in the statement section.

The FORTRAN procedure form (Figure 3.5) has the same general format as the other forms. FORTRAN statements are entered in the statement section. Standard FORTRAN assignment, GO TO, arithmetic IF, and continue statements are entered in the statement field, with statement numbers entered in the number field. Variables and parameters are automatically posted to the variable and parameter tables as the FORTRAN statements are entered.

When the user finishes describing the model, the model description is processed by the BIOMOD compiler. The resulting output is then compiled by the CSMP and FORTRAN compilers. The resulting object modules are linked and loaded, and the



STORE 1 RECALL FRAME 008313	LOG ON	MODEL EDITOR	HARD COPY	ROLL
	START OF MODEL	PROCESS LIST	DATA	
	PRIM FORTRAN	PROC EX SEC.	DATA	PAGE
FIGURE				ORIGIN
ENTRY				

SCROLL

FORTRAN PROCEDURE CODING FORM

```
NUMBER     STATEMENT

              Y = 5

              IF (X - X1) 99, 91, 10
  10          IF (X - X2) 91, 90, 20
  20          IF (X - X3) 90, 99, 99

  90          Y = 0
              GO TO 99

  91          Y = 1

  99          CONTINUE
```

VARIABLES

CHECK IF:
PLOTTABLE
NOT PLOTTABLE

PLOTTABLE	NOT PLOTTABLE	NAME	COMMENT
		Y	
		X	

PARAMETERS

SCROLL

CHECK IF:
MODIFIABLE
NOT MODIFIABLE

MODIFIABLE	NOT MODIFIABLE	NAME	VALUE	COMMENT
		X3	+1.0000E+00	
		X2	+1.0000E+00	
		X1	+1.0000E+00	

Figure 3-5 BIOMOD FORTRAN procedure form

simulation started. The compiling and loading is automatic. The user is kept posted on the state of the system during compilation, and any detected errors cause termination of compilation and return to the user with appropriate error messages.

During simulation the variables indicated by the user are graphed on the interactive device's display. By depressing the attention key, the user may stop the system at any time and change model parameters and examine the state of model variables. The simulation may the either be halted or restarted.

3.2.3 <u>System Structure</u>. The BIOMOD system is divided into three sections as shown in Figure 3.6. The construction phase deals with the building of the model description. The compilation phase encompasses the translation and compilation of the model description into a machine-level computer program. The simulation phase covers the running of the program and displaying of simulation results.

The construction phase is largely derived from the Rand Grail programs for man-machine interaction [Ellis, Heafner, and Sildey 1969]. The model description, as outlined in section 3.2, is translated into ring structures. This allows rapid access to the various parts of the model description for display and later use in the compilation phase. This description resides on secondary random-access storage, and is cross-indexed for rapid storage and retrieval.

The compilation phase involves the BIOMOD compiler, CSMP program, FORTRAN compiler and system linkage/editor programs as

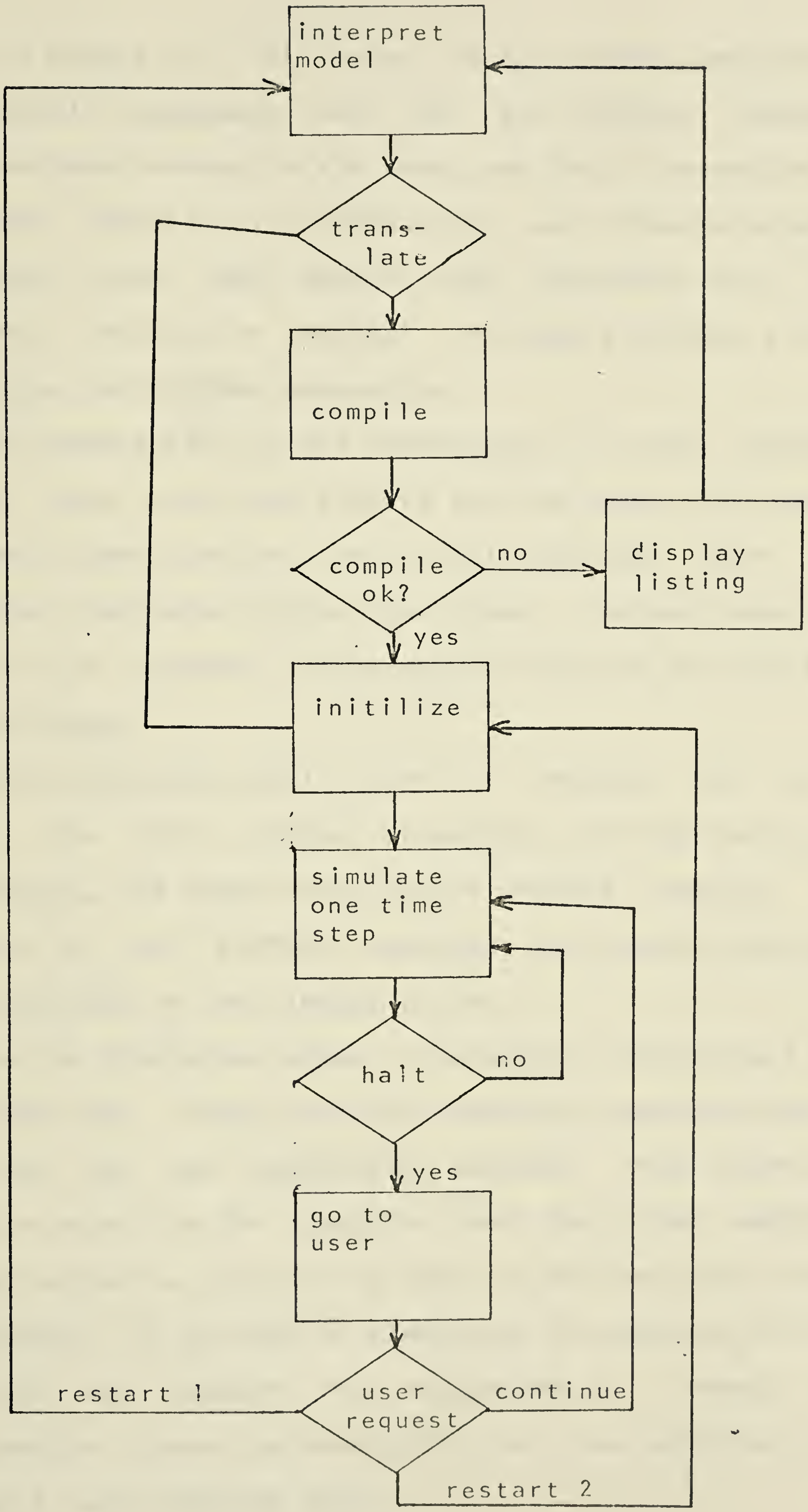
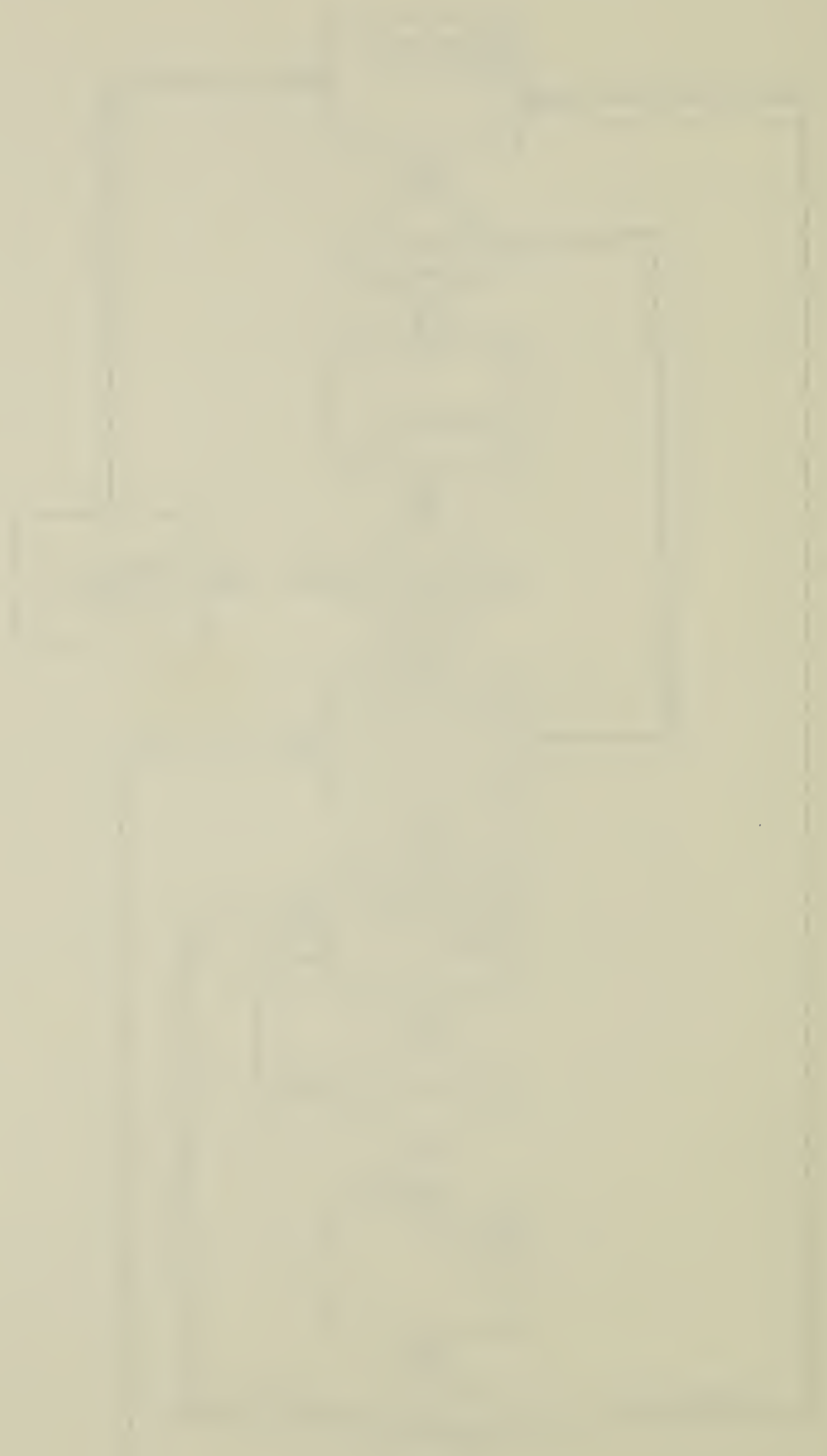


Figure 3-6 BIOMOD system flow of Control



shown in Figure 3.7. The output of the BIOMOD compiler consists of control statements and CSMP and FORTRAN programs that describe each process in the model and their interrelationship. Processes defined by mathematical and chemical equations are translated into CSMP macros that represent the process. Processes defined by FORTRAN statements produce a CSMP macro that calls the FORTRAN subroutine.

The description of the interaction of these processes is derived from the block diagram for the model. For each level, the named boxes produce a macro call using the box's name and parameters indicated by the flow lines. Unnamed boxes cause the compiler to continue examining the diagram for named boxes on the next level.

This continues until a level is reached with no unnamed boxes. The CSMP program translates the CSMP macros, FORTRAN subroutines, and macro calls into a FORTRAN program. This is compiled by the FORTRAN compiler and linked with simulation phase programs by the linkage editor.

In the simulation phase, interactive routines are called at each time step. These routines establish communication between the user and the simulation program. They allow access to variable locations for graphing and run time modification. Graphic output is provided by reactant and variable values.

Control at the end of simulation is returned to the user. The user may restart the simulation or return to the construction phase for description of a new model or structural changes to any existing model.

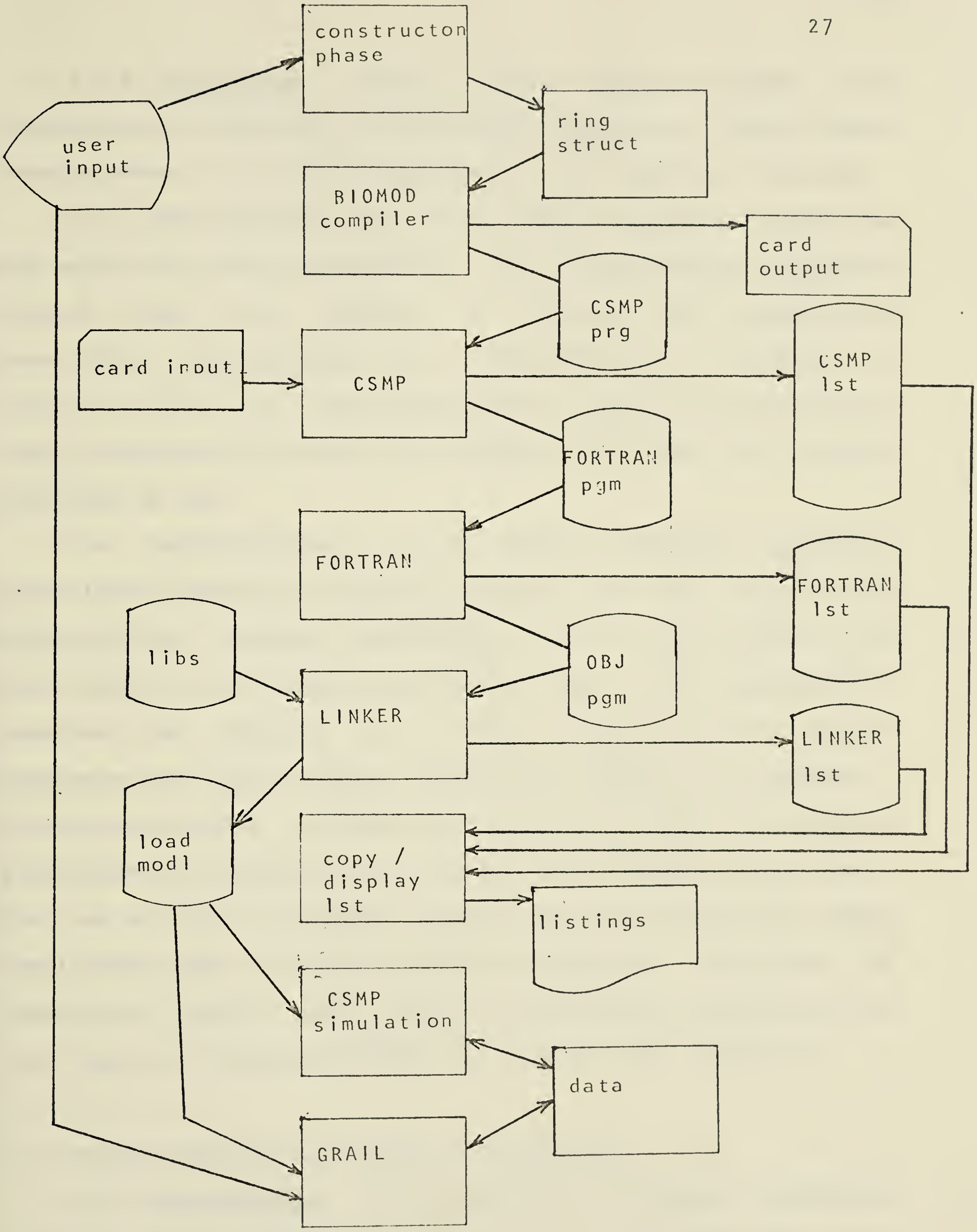


Figure 3-7 BIOMOD System Flow of Data

[illegible]

3.2.4 Evaluation. BIOMOD offers several methods for integration of the model's differential equations. All of these methods exceed the current requirements for numerical accuracy.

The most appealing part of the BIOMOD simulation system is the method for model description. The compartmental approach coupled with the ability to describe each compartment chemically, algebraically, or algorithmically produces a powerful means for model description. The use of interactive input, with the Rand Tablet and graphics CRT, makes the system very easy to use.

The implementation is not easily portable, requiring specialized interactive hardware such as the Rand Tablet and sophisticated graphics facilities. This would reduce the availability of the system for general use. The architecture requires the resources of a large computer for the various subsystems and could require considerable effort to implement. The current version of BIOMOD runs on an IBM 360/65 and requires a partition of 228,000 bytes of memory with overlays from disk. The use of both the FORTRAN compiler and CSMP program for model compilation make the system somewhat cumbersome. Correction of compilation errors could prove to be difficult because of the many layers of programs between the user and the simulation.

3.3 Kinetic Simulation Language for Chemistry

3.3.1 Introduction. Work was begun on a digital simulation system by Britton Chance in 1953 when it was realized that then existing analogue computers were insufficient to simulate systems that were complex enough to be of interest [Garfinkel

1968]. The resulting programs have undergone many changes and revisions to improve the accuracy, speed, and ease of simulation.

The design philosophy was to create a series of digital computer programs that would allow the user to describe the chemical system under study in a more representative form than the differential equations used in analogue simulation. This necessitated the development of a language to describe the chemical system, and the writing of a compiler for the input language.

The most recent version of this language was developed by David Garfinkel of the Johnson Research Foundation. Garfinkel's "Kinetic Simulation Language for Chemistry and Biochemistry" [Paterson and Garfinkel 1968] is a digital simulation system. There is no attempt to mimic an analogue computer at any level of the simulation system. The system is batch-oriented, and requires the use of a FORTRAN compiler in the production of the object program. Input to the system is fairly straightforward, consisting of chemical flux equations and numerical data such as reactant concentrations and rate constants. Additional input controls simulation parameters such as the degree of numerical accuracy, and form and amount of output from the simulation run [Garfinkel 1968].

The system is fairly machine independent, being written in FORTRAN, and can be run on most moderate sized digital computers. The present system runs on the IBM 360 series computers, and requires 90,000 bytes of storage. Slight modification to the I/O routines may be necessary for

implementation on othe machines. It is easy to use and produces simulations within a range of accuracy of .1% [Garfinkel 1968, Chance, Chance, and Seller 1960].

3.3.2 Language Description. The chemical system to be modeled is described by chemical flux equations on cards or card images. The card format is:

col.	1	statement type identification code
col.	2-5	statement numbering
col.	6	continuation card indicator
col.	7-72	model statement
col.	73-80	sequence numbering

The chemical flux equations consist of reactant expressions alternating with operators. A reactant expression is defined as a reactant name which may be preceded by a numeric giving the reactant's concentration. Reactant names are contiguous strings of up to 20 alphanumeric characters beginning with an alphabetic character. The operators are +, -, and =, with one or more spaces serving as delimiters. The + separates up to 12 reactant expressions in each half equation. The - seperates half equations in non-reversible reactions. The = separates half equations in reversible reactions. The end of the card(s), a '/', or a '(' terminates an equation. The / may be followed by the rate constant(s) in decimal or E format. Fluxes may follow the (in the same formats.

Stoichiometrics are given on separate cards following the equation to which they apply. The format is:

col. 1	S
col. 77-26	reactant name, left justified
col. 30-72	stoichiometric in decimal or E format. It may take any real value, even 0.

Reactants in an equation may have their concentrations unaltered by the equations flux, as in the case of a catalyst, by specifying them as constant reactants. Constant reactants are given on separate cards following the equation in which they occur. The format is:

col. 1	H
col. 2	U, holds concentration constant for all equations in system, if blank the reactant's concentration is constant only in the preceding equation.
col. 7-26	reactant name, left justified.

The last equation is followed by an end sentinel card with an E in col. 1. This marks the end of the model description. Following the end sentinel card are control and numerical parameter cards. Control cards specify the amount and form of output from the simulation run, character sets, and other operational parameters.

The general format is:

col. 1	identifying character
col. 2-5	equation number where applicable
col. 7-26	reactant name, key characters /, (, or numeric value
col. 30-39	numeric value
col. 40-72	numeric value for the time the specification is to take effect

Any values not given in the model description or later specified on numerical parameter cards are set to 0.

These cards are followed by another end sentinel card with a Z in col. 1. Decks may be stacked, each followed by the Z end sentinel card. The final card for the entire deck is a Z end sentinel card with a Z also in col. 2. For a full description of the language see [Higgens 1965] and [Larson, Seller, and Meyer 1962]. An example of a typical simulation deck is given in Figure 3.8 [Garfinkel 1968; Paterson and Garfinkel 1968].

```
T            SAMPLE OF SIMPLE ENZYME SYSTEM

C title for listing
             E + S = ES

C first equation
             1/1.0E8   /1.0E3

C rate constants for first equation
             ES = E + P

C second equation
S            P       2.0

C stoichiometric for P in second equation
E
C indicates end of equations
           3 /       1.0E-1

           4 /       1.0E8

C rate constants for second equation
             E       1.0E-1

             S       1.0

             ES      0.03

C initial reactant concentrations
LABEL        RUN1

C label for run
TIME         1.0E-3

C minimum time step
ERROR        1.0E-3

C error bound
G            E       1.0E-3   E
G            ES      1.0E-3   C

C graph instructions
Z
ZZ
C signifies end of deck
```

Figure 3.8 Sample simulation deck for Garfinkel's system

3.3.3 <u>System Structure</u>. Garfinkel's simulation system is a batch-oriented, compiled system. Three job steps are needed for each complete simulation run. This requires the use of intermediate file storage between job steps, with the associated job control language. This is an unfortunate burden the user must bare. The first two steps produce the object module, which is linked and loaded and run in the third step (Figure 3.9).

The simulation system decomposes into three logical parts, coinciding with the three job steps. These are: the generator section which makes the first pass of the input stream; the FORTRAN compiler which compiles the output of the generator section; and an operating section, which when linked with the output from the FORTRAN compiler and loaded, produces the final simulation output.

The generator section takes the source language as input and outputs two FORTRAN source code routines. The first subroutine, DERV, consists of a description of the system in the form of a set of differential equations representing the first derivatives of the concentrations of the reactants in the system. The second routine, MAIN, is the mainline program for the simulation. It contains the initial conditions, vectors containing reactant concentrations, fluxes, other model data, and calls to the routines that produce a stepwise integration of the equations in DERV.

The output from the generator is compiled by the FORTRAN compiler. The two resulting object modules represent the final machine code description of the model. These are linked with the operating program to produce a complete model simulation

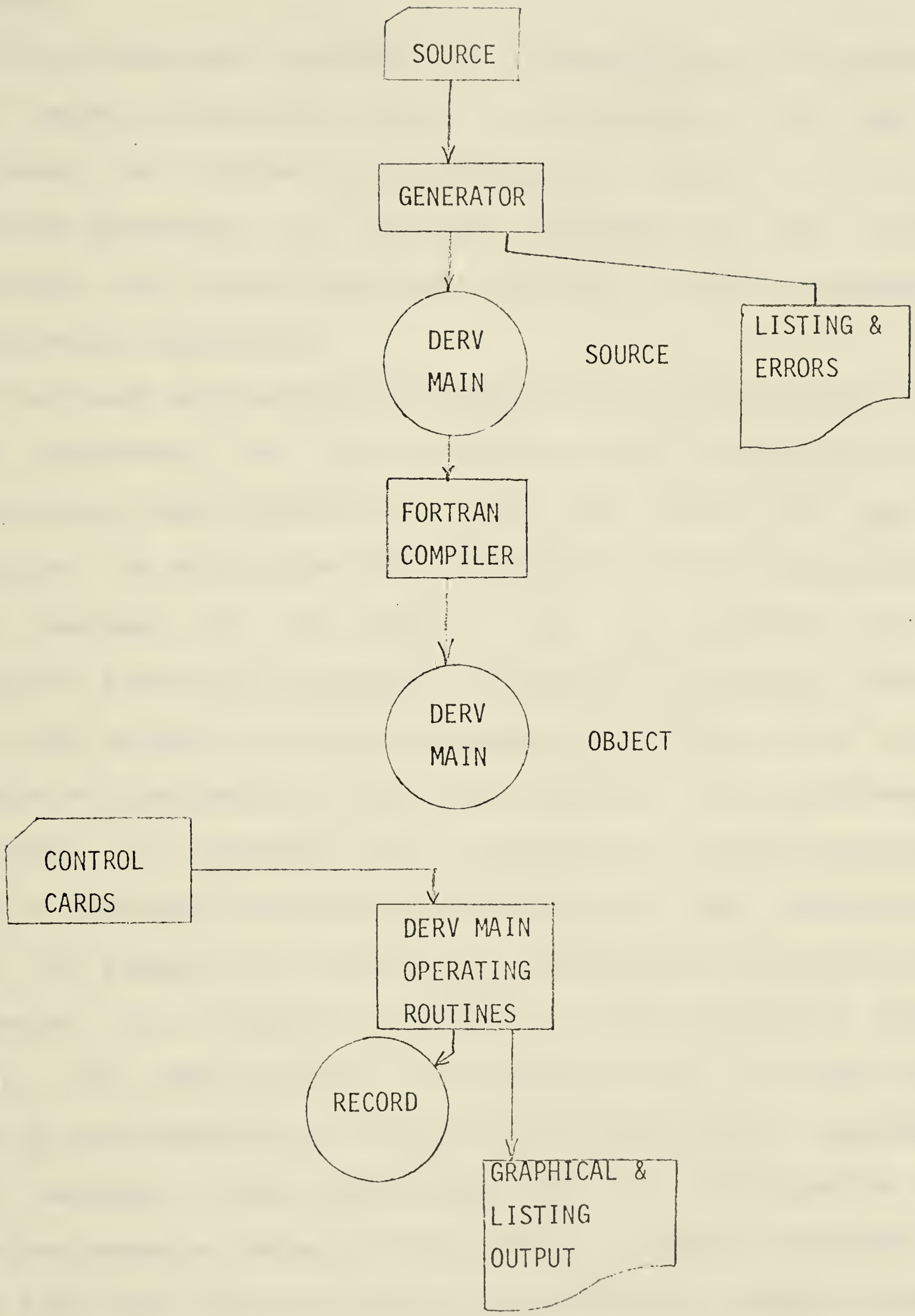


Figure 3-9 System structure

program.

The generator program is straightforward in operation. Each input statement's type is determined, and then the statement is scanned for syntactical errors. If it is an equation statement, it is then processed by the following algorithm for converting flux equations to their corresponding differential equations:

For each half equation, multiply the concentrations of all the reactants in half reaction being processed by the appropriate rate constant to yield the flux for that half reaction. To obtain the time derivative of the concentration of each reactant in the system, form the algebraic sum of the fluxes of those half reactions involving the chemical, where the sign used is plus or minus depending upon whether the reactant appears or disappears in the half reaction. If a stoichiometric (S card) is specified for a reactant in a half reaction, the flux of that half reaction is multiplied by the stoichiometric when it appears as a term in the differential equation of that reactant. If a reactant is named on a 'hold constant' card (H card), its concentration is multiplied into the flux, but the flux is not included as a term in the differential equation of the reactant. This effectively holds the concentration of the reactant constant through time.[Larson, Seller, and Meyer 1962]

After the input statements are processed, storage areas are set aside for reactant concentrations, first and second derivatives, and other data necessary for the solution of the differential equations.

Any control cards are read, scanned for syntax and

processed into the corresponding table entries that specify the simulation parameters and control output routines. The two resulting FORTRAN source routines, DERV and MAIN, are then written out to a user-specified intermediate storage device.

These routines are compiled by the FORTRAN compiler, linked with operating program, and loaded. After reading and processing any control cards, the operating program proceeds in a first order Euler method solution of the differential equations that represent the system. The following algorithm is used. The integration step size, DT is variable and is initially set a user-specified value STINT. DT is always subject to the constraint

$$\text{minimum}\ (.11920928\ \text{E-6},\ \text{STINT}) \leq \text{DT} \leq \text{XMINT}$$

where XMINT=.02 TIME (for TIME>0.0).

DT assumes the value of the bound if its calculated value exceeds the bound.

The model is then simulated using the following algorithm.

1. $n = n + 1$
2. compute DT by

   $$\text{DT} = \text{error} + \max\left[y\ (i,n)/y''(i,n) \right]$$

   where $Y(i,n)$ is the concentration of the ith reactant in the system at time n.
3. Compute the first derivative for each reactant.
4. Compute new reactant concentration at timestep n as

   $$y(i,n+1)=y(i,n)+y'(i,n)\ *\ \text{DT}$$
5. calculate the second derivatives of each reactant as

   $$Y''(i,n) = [Y'(i,n) - Y'(i,n-1)]/\text{DT}$$

6. time = time+DT

7. If the time limit has been reached, exit, otherwise Go back to step 1.

Data is output, or stored for output, at every iteration if requested by the user. The time is also monitored so that any reactant concentration changes, or other system changes specified on control cards, are performed at the appropriate time during the integration.

The final output of the simulation run consists of system dumps, either to the printer or some mass storage device, and printer-plots of the concentrations of up to six reactants. The system dumps are the values of the system's parameters (such as reactant concentration, first derivatives, DT, TIME, etc.) presented in a labelled tabular format [Garfinkel 1968, Garfinkel, Rutledge, Higgens 1961, Larson, Seller, Meyer 1962].

3.3.4 Evaluation. David Garfinkel's simulation has several good points. Although the numerical method for the solution of the differential equations is crude, it is simple, which enhances speed, and will produce a solution within 0.1% degree of accuracy [Chance, Chance, and Seller 1960].

The use of flux equations to describe the structure and kinetics of the system is simple and straightforward. Unfortunately, the specification of a stoichiometric by using a separate card and a concentration as a numeric prefix for a reactant name is confusing.

The stoichiometric should be the prefix, and the concentrations expressed in a more standard form, such as:

```
          GLOB30 = 1.70 E-3
          or more simply
          GLOB30 = 0.0017
          or both.
```

The ability to hold the concentration of any reactant constant in a reaction of the whole system is good. It would also be nice to treat any reactant that appears on both sides of the flux equation with the same stoichiometric as a "hold constant" reactant. This reactant's concentration will not be affected by the flux of the equation. There is no need for the equation's flux to appear in the expression for the first derivative of the reactant's concentration.

Although the simulation language is fairly straightforward and complete, the simulation system's structure detracts strongly from the total usefulness of the simulation system. The use of the FORTRAN compiler is largely responsible for this.

The user faces a batch-oriented, three job step, simulation run. This could be corrected by writing an interactive monitor that would generate the job steps necessary to run a simulation, and handle a more format-free input language. But this would not correct the rather inflexible nature of the system.

To change the model, the user must edit the input with a text editor, restart the simulation system, recompile and rerun the simulation. This is time-consuming and laborious. An interpretive approach, if the speed of the integration can be maintained, would correct this. There would be no need to

recompile the model after each change. Retrieval and editing of the model description would be easily handled within an interpretive system.

Garfinkel's system has a clear and reasonably accurate language for model description. His numerical methods are excellent and produce numerical accuracy within 0.1%. Unfortunately, the system's structure makes it seem cumbersome to use, and strongly detracts from its usefulness to the user.

4. THE INTERACTIVE BIOCHEMICAL SIMULATION SYSTEM

4.1 Introduction

The final system to be discussed is the Interactive Biochemcial Simulation System (IBSS). IBSS was developed to provide an interactive means of simulating complex biochemical systems on a digital computer. IBSS permits the determination, as a function of time, of the properties of any complex system defined by chemical reactions and associated numerical data, where the behavior of the chemical equations follows the law of mass action. The system was designed to be capable of simulating rather complete biological phemenon such as gene expression and assembly in bacteriophages, or operon control systems in bacteria. Existing systems were judged to be either inadequate or to require too extensive an effort to implement simulations of such scope.

IBSS is designed to provide an interactive simulation system that would be easy to operate and employ an intuitive format for formulation and development of biochemical models. A model is described by chemical flux equations and numerical data. There are facilities to automatically monitor and alter the state or structure of the model during simulation. The user can monitor the simulation and dynamically alter the model's state and structure during simulation or formulation. Models and simulation results can be stored, reviewed, and combined. This makes it possible to develop and study subsets of the desired model separately. These subsets may later be combined and studied as a unit.

The macro features of the system, which enable dynamic modification of the model, were designed to facilitate the modeling of complex control processes likely to be encountered when modeling an organism. While it is possible to model such control processes solely in terms of flux equations, doing so can result in a large set of equations. By employing a set of macro commands that permit the regulation and alteration of the flux equations, it is possible to model control functions in a more direct fashion.

IBSS is written in FORTRAN and, with the exception of the input-output modules, is machine independent. It is designed as a hierarchical set of modules communicating through a common set of data modules. This facilitates modification of the system to tailor it to the computer configuration limitations or advantages. On large machines, such as the IBM 360/67 at the University of Alberta, all the operating modules and data storage areas would be resident, while on smaller machines such as a 100K PDP 11/45, they can be kept on a secondary storage device as overlay structures, and brought into main memory as needed.

The response time is reasonable. The model in Appendix A was simulated on an IBM 360/67 under the Michigan Terminal operating system for 20 time steps. This required only 18.839 seconds of CPU time with an overall execution time of 1.85 minutes. The accuracy of this simulation was within 0.1%. The run listing and computation of accuracy are given in Appendix A.

4.2 Language Description

4.2.1 CRT Monitor. The model is described by a set of chemical flux equations and related numerical data. The language will be described in the context of a remote CRT/Keyboard display terminal with local memory. The displays shown in Figures 4.1 through 4.5 show a model for allosteric (shape changing) inhihition for a simple enzyme system. This is the same system discussed in Chapter 1. Two methods for modeling this system are shown. One uses only flux equations, while the other uses macro commands for control purposes.

The input format is a series of video displays with writable areas for entering the model description and other data necessary for simulation of the model. The first display the user sees is the status display (Figure 4.1). This display gives the file name for storage and/or retrieval of the model. The present time in the simulation run and maximum time limit are given, with the error bound, command status (CTAG), minimum integration step size, and number of data sample points to be saved. These would be specified by the user. There is an area for messages to be logged from the simulation system to the user. These are effectively scrolled.

The user indicates the next display he wishes, or action to be taken by the system, under the 'Commands' section of the display.

```
+-----------------------------------------------------------------------+
| 11:59:00                        IBSS                        14OCT75   |
+-----------------------------------------------------------------------+
| INDICATE COMMAND WITH *                                               |
+----------------+------------------------------------------------------+
| _ DISPLAY      | _EQUATIONS   _REACTANTS         _GRAPH               |
| _ ALTERATION   | _STATUS      _INTERNAL CMDS.    _STOP                |
|                | _RUN         _DERIVATIVES       _INSTRUCT            |
|                | _PDUMP       _DUMP              _SAVE                |
|                | _MERGE       _DESTROY                                 |
+----------------+------+-----------------------+-----------------------+
| _ SCROLL FORWARD      |                       |   _SCROLL BACK        |
+-----------------------+                       +-----------------------+

     SYSTEM STATUS:

          PROGRAM FILE: SIMPAL-SYSTEM

     RUN STATUS:

          TIME 13.127       MAXTIME 20.00    ERROR 1.0 %

          MINSTEP 0.01      # SAMPLE POINTS 40

          COMMAND STATUS 1

     MESSAGES:
```

Figure 4.1 IBSS monitor display

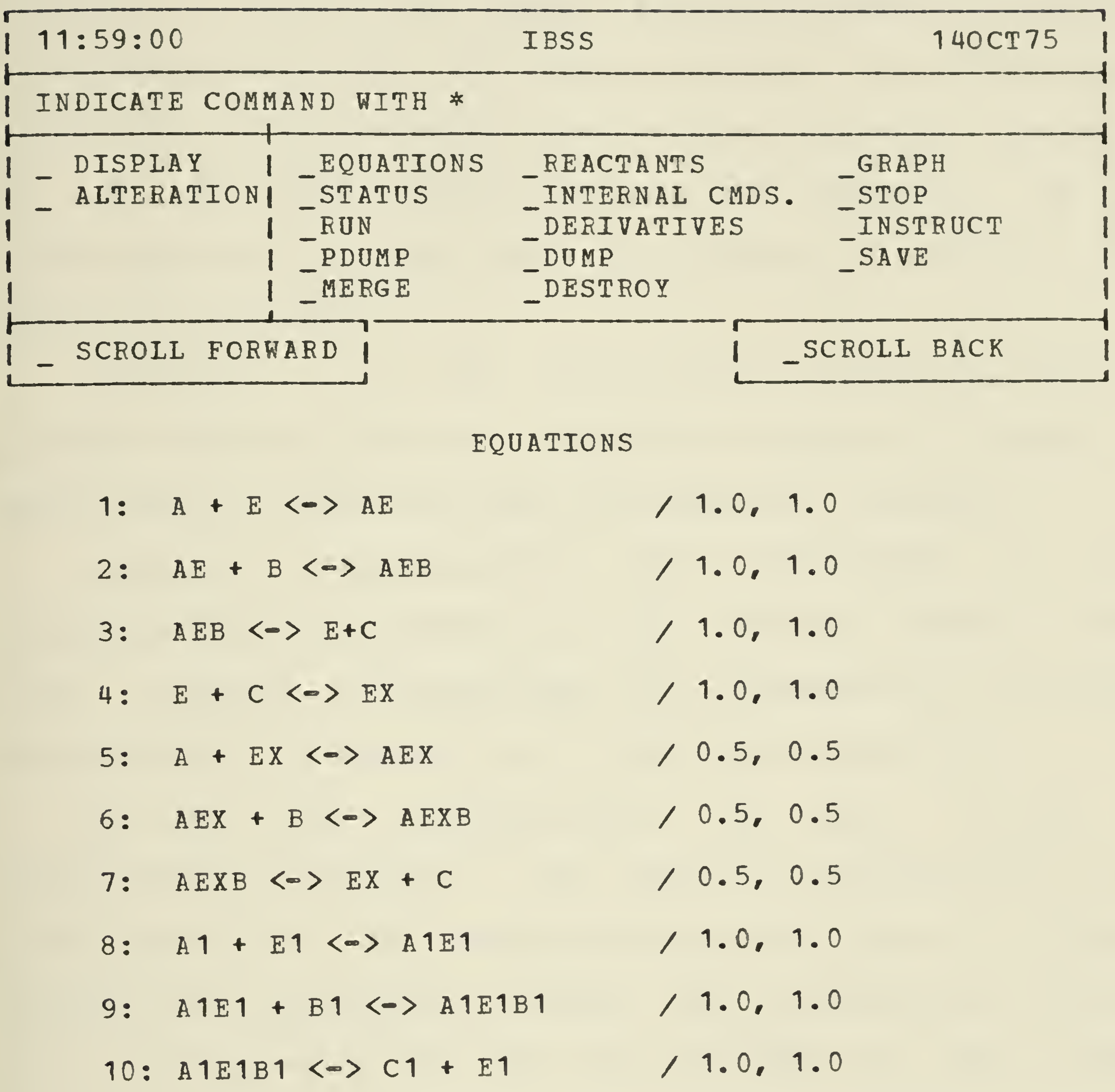
The Commands are:

```
Graph       - present graph display
Destroy     - delete   the   program   file   given   under
              'Program File'
Derivatives - present derivative display
Equation    - present equation display
Stop        - return to operating system
Run         - produce a simulation of the model
Merge       - combine the present model with the one  in
              the file given under 'Program File'
Instruct    - present an instruction display
Pdump       - print a 'snapshot' of the model
Save        - store the model in the program file
Dump        - print  a complete description of the model
              and simulation
Reactants - present the reactants display
Internal Commands
            - present the internal commands display
```

The equation display (Figure 4.2) is used to enter the chemical flux equations that describe the model. The equations consist of a unique equation number followed by a colon, two half reactions separated by a ---> for non-reversible equations, or a <--> for reversible equations. The equation is followed by a /, the forward rate constant in integer, decimal, or exponential format, and where applicable the reverse rate constant, preceded by a space or comma.

Each half equation is composed of one or more reactant terms separated by +. Each reactant term has a stoichiometric and reactant name. The stoichiometric is optional and when not given is set to 1.0. When specified it is a numeric in decimal or integer format preceding the reactant name. The reactant name must appear and is made up of a contiguous string of from 1 to 12 alphanumeric characters beginning with an alphabetic character.

In Figure 4.2 equations 1 through 3 show the progression from substrates A and B and enzyme E through intermediates AE



```
11:59:00                         IBSS                       14OCT75

INDICATE COMMAND WITH *

_ DISPLAY      _EQUATIONS   _REACTANTS          _GRAPH
_ ALTERATION   _STATUS      _INTERNAL CMDS.     _STOP
               _RUN         _DERIVATIVES        _INSTRUCT
               _PDUMP       _DUMP               _SAVE
               _MERGE       _DESTROY

_ SCROLL FORWARD                                _SCROLL BACK

                          EQUATIONS

    1:  A + E <-> AE                   / 1.0, 1.0

    2:  AE + B <-> AEB                 / 1.0, 1.0

    3:  AEB <-> E+C                    / 1.0, 1.0

    4:  E + C <-> EX                   / 1.0, 1.0

    5:  A + EX <-> AEX                 / 0.5, 0.5

    6:  AEX + B <-> AEXB               / 0.5, 0.5

    7:  AEXB <-> EX + C                / 0.5, 0.5

    8:  A1 + E1 <-> A1E1               / 1.0, 1.0

    9:  A1E1 + B1 <-> A1E1B1           / 1.0, 1.0

   10: A1E1B1 <-> C1 + E1              / 1.0, 1.0
```

Figure 4.2 IBSS equation display

and AEB, finally resulting in the product C and free enzyme E. Equation 4 shows free enzyme E being modified by product C, forming the enzyme EX. Equations 5 thorugh 7 are similar to the first three equations except they envolve enzyme EX instead of E, and have lower values for their rate constants. The first seven equations represent modeling allosteric inhibition using only flux equations.

Equations 8, 9, and 10 are used in conjunction with the reaction monitors and macro commands in figure 4.2 to model the same series of reactions given in equations 1 thorugh 7.

Reactant concentrations and constraints on their concentrations are specified in the reactant display (Figure 4.3). Reactant concentrations are displayed by the reactant name enclosed in brackets, an =, and the concentration.

To set the concentration, the user changes the concentration displayed. The default concentration is set initially to 0.0. The concentration may be given in integer, decimal, or exponential format. The concentrations displayed are the concentrations of the reactants at the present simulation time. Reactants may have their concentrations held constant by specifying the reactant name under "Buffered Reactants." This is often useful in mimicking the replenishing of substrate pools in the model by functional units of the system that are not included in the model description. Reactants may have their concentrations increased or decreased at a fixed rate by specifying the reactant name, followed by a / and the rate of change under the 'External Input' heading. This can be used for many purposes, one of which is shown in the

```
┌─────────────────────────────────────────────────────────────────────┐
| 11:59:00                          IBSS                     14OCT75  |
├─────────────────────────────────────────────────────────────────────┤
| INDICATE COMMAND WITH *                                             |
├────────────────┬────────────────────────────────────────────────────┤
| _ DISPLAY      | _EQUATIONS   _REACTANTS         _GRAPH             |
| _ ALTERATION   | _STATUS      _INTERNAL CMDS.   _STOP              |
|                | _RUN         _DERIVATIVES       _INSTRUCT          |
|                | _PDUMP       _DUMP              _SAVE              |
|                | _MERGE       _DESTROY                              |
├────────────────┴─────┐                        ┌─────────────────────┤
| _ SCROLL FORWARD     |                        |  _SCROLL BACK       |
└──────────────────────┘                        └─────────────────────┘

                              REACTANTS

     CONCENTRATIONS:

      [A] = 4.0     [B] = 2.0     [C] = 0.0     [E] = 4.0

     [A1] = 4.0    [B1] = 2.0    [C1] = 0.0    [E1] = 4.0


     EXTERNAL INPUT:

       C / -1.0

      C1 / -1.0


     BUFFERED REACTANTS:
```

Figure 4.3 IBSS reactants display

model example in Chapter 5.

The derivatives display (Figure 4.4) presents the first derivative values for reactants at the present simulation time. The first derivative expansion of the model for each reactant may be assembled and displayed upon request. The reactant names are entered after the First Derivative Expansion heading. The simulation system will then assemble the derivative expression, and/or the derivative value.

The macro commands display (Figure 4.5) is used to enter statements that monitor the state of the simulation and automatically alter the state or structure of the model. The reactant monitor statements are checked at each integration step, and if any of the conditions specified is met, the command status is set to the specified value and execution is transferred to the specified macro command. The monitor statements are evaluated in the order that they are entered. Their format is a 1 to 4 alphanumeric statement label, space or spaces, then two variables separated by an operator (.LT.; .GT.; .LE.; .GE.; .EQ.; .NE.), space or spaces, the value to the assigned to the command status if the condition is met, a comma, and the macro name to transfer execution. Variables in the conditional part of the statement may be reactant names, representing the value of their concentrations; reactant names enclosed in parentheses and preceeded by a 'd', representing the value of their first derivative; decimal, integer, or exponential numbers: CTAG for the current status value, DT for the value of the time step, and TIME for the current simulation time.

```
┌──────────────────────────────────────────────────────────────────┐
| 11:59:00                      IBSS                      14OCT75  |
├──────────────────────────────────────────────────────────────────┤
| INDICATE COMMAND WITH *                                          |
├───────────────┬──────────────────────────────────────────────────┤
| _ DISPLAY     | _EQUATIONS   _REACTANTS        _GRAPH            |
| _ ALTERATION  | _STATUS      _INTERNAL CMDS.   _STOP             |
|               | _RUN         _DERIVATIVES      _INSTRUCT         |
|               | _PDUMP       _DUMP             _SAVE             |
|               | _MERGE       _DESTROY                            |
├───────────────┴────┐                    ┌────────────────────────┤
| _ SCROLL FORWARD   |                    |  _SCROLL BACK          |
└────────────────────┘                    └────────────────────────┘

                         DERIVATIVES

    FIRST DERIVATIVE VALUES: A    E1

    D[A] / DT = 1.63 E-1

    D[E1] / DT = 1.7368 E-2


    FIRST DERIVATIVE EXPANSIONS: E1

    D[E1]/DT=1.0[A1][E1]+1.0[A1][E1][B1]-1.0[A1][E1]-
    1.0[C1][E1]
```

Figure 4.4 IBSS derivatives display

The reaction monitors may serve many purposes. They can be used to monitor the simulation and halt it if things get really out of hand. They may also be used to check for maximum or minimum points in the concentrations of the reactants, by checking for sign changes in the first derivatives of the reactants.

The reaction monitors may also be used to sense threshold conditions for macro-coded regulation of the model. This is the use of the two reaction monitors in Figure 4.5. The first reaction monitor test if the concentration of reactant C1 is greater than 5.5 concentration units. If so, the simulation is temporarily halted and macro statment M1 is executed. Similarly the second reaction monitor checks for concentrations of reactant C1 less than 4.5. When this condition occurs, the simulation is temporarily halted and macro statment M10 is executed. These two reaction monitors establish the thresholds for the allosteric inhibition control being modeled.

The macro commands allow automatic modification to the state and structure of the model. Reactant concentrations can be changed, equations altered in any way, the simulation restarted or stopped. A complete list of macro commands is given in Appendix B. A detailed example of their use is given in Chapter 5. The format for the macro commands is a four-character name as in the monitor statements, a space or spaces, the mnemonic for the command, a space or spaces, and the operands for the command separated by commas.

A brief example of the use of macro commands for modeling control mechanisms is shown in Figure 4.5. The macro command M1

```
+----------------------------------------------------------------------------------+
| 11:59:00                         IBSS                              14OCT75       |
+----------------------------------------------------------------------------------+
| INDICATE COMMAND WITH *                                                          |
+------------------+---------------------------------------------------------------+
| _ DISPLAY        | _EQUATIONS    _REACTANTS           _GRAPH                     |
| _ ALTERATION     | _STATUS       _INTERNAL CMDS.      _STOP                      |
|                  | _RUN          _DERIVATIVES         _INSTRUCT                  |
|                  | _PDUMP        _DUMP                _SAVE                      |
|                  | _MERGE        _DESTROY                                        |
+------------------+--------+                   +----------------------------------+
| _ SCROLL FORWARD          |                   |  _SCROLL BACK                    |
+---------------------------+                   +----------------------------------+

                              MACROS

REACTION MONITORS:

     C1  .GT.  5.5    1,M1

     C1  .LT.  4.5    1,M10


MACROS:

     M1:    INCR   FRATE (8, 10), -0.4

     M2:    INCR   BRATE (8, 10), -0.4

     M3:    CONT

     M10:   INCR   FRATE (8, 10), 0.4

     M11:   INCR   BRATE (8, 10), 0.4

     M12:   CONT
```

Figure 4.5 IBSS macros display

will be executed whenever the threshold condition specified by the first reaction monitor in Figure 4.5 is met. Macro statment M1 increments the forward rate constants for equations 8 through 10 by -0.4. The macro statment M2 is then executed, and increments the reverse rate constants for equations 8 through 10 by -0.4. Macro statment M3 continues the simulation.

In a similar fashion macro statments M10 through M12 are executed whenever the threshold conditions in the second reaction monitor are met. Macro statments M10 and M11 increment the forward and reverse rate constants for equations 8 through 10 by 0.4. Macro statment M12 continues the simulation.

At present there is a limit to 30 reaction monitors and 100 macro commands per model in IBSS. This is a system parameter and can be easily be changed. The reaction monitors and macro commands can be used to model control mechanisms, check for error conditions, or automatically vary the model's constraints and/or structure to adjust the model's behavior to agree with known data. At present the commands are a bit terse, but they provide a broad range of manipulative functions for model control and regulation.

Graphical representations of reactant concentrations are presented in the graph display. Up to six reactants may be graphed at a time. The axes are adjusted to the constraints specified by the user in the display. Hard copy of the graph is available upon request by the user (Figure 4.6). To run the system, the user simply runs the program, enters his model description using the various displays, or reads in the previous model description. The user could review previous simulation

```
+------------------------------------------------------------+
| 11:59:00                   IBSS                  14OCT75   |
+------------------------------------------------------------+
| INDICATE COMMAND WITH *                                    |
+--------------+---------------------------------------------+
| _ DISPLAY    | _EQUATIONS  _REACTANTS       _GRAPH         |
| _ ALTERATION | _STATUS     _INTERNAL CMDS.  _STOP          |
|              | _RUN        _DERIVATIVES     _INSTRUCT      |
|              | _PDUMP      _DUMP            _SAVE          |
|              | _MERGE      _DESTROY                        |
+--------------+---+                    +-------------------+
| _ SCROLL FORWARD |                    |  _SCROLL BACK     |
+------------------+                    +-------------------+

                             GRAPH

    CONC

    7.0 +
        +
    6.0 +
        +
    5.0 +
        +
    4.0 +E E E                             C C
        +     E E                      C
    3.0 +         E E            C  C
        +              E E C  C
    2.0 +             C  C  E E
        +     C  C              E  E
    1.0 +   C                         E E
        + C                               E E
    0.0 ++++++++++++++++++++++++++++++++++++++++

        0         5        10        15        20

                              TIME

    REACTANT : SYMBOL   TIME SPAN 0.0 TO 20.0

       C : C            CONC.  SPAN 0.0 TO 7.0

       E : E
```

Figure 4.6 IBSS graph display

results, or set the necessary numerical parameters and run a new simulation. The results of the simulation run could be reviewed and possibly saved for later analysis.

4.2.2 Teletype Monitor. A teletype monitor has been implemented. Rather than using formatted displays as in the CRT monitor, commands followed by text strings are used. The text strings give the necessary information for the command, with commas used to delimit each element of the command's text string. For example to input an equation the user types:

```
INPT   F1   E + S<-->ES  /1.0, 1.0
```

which enter equation F1. To get a graph of the concentrations of E and S from time 0 to 20, the user enters

```
GRPH   0.0, 20, E, S
```

A list of the teletype monitor commands is given in Appendix C. These commands are grouped into six categories according to their function:

1)	File Management	- commands for the storage and retrieval of the model to the mass data device.
2)	Model Description	- commands for input of equations, concentrations and reactant constraints, and macro commands
3)	Execution	- commands for the actual simulation process.
4)	Retrieval	- commands for requesting information about the model or macros.
5)	Model Manipulation	- commands for modification of the model's state or structure.
6)	IOMON Local COmmands	- commands performed internally in the I/O monitor.

The commands provide full use of the IBSS facilities. They are a bit terse, but were designed for testing the functions of the

system, and not as user-oriented monitor.

A sample of a session with IBSS using the teletype I/O monitor is given in Figure 4.7. In this figure the model

$$E + S \longleftrightarrow ES$$

$$ES \longrightarrow P + E$$

is simulated for 20 time steps with initial concentrations [E] = 1.0, [S] = 8.0, [ES] = 0.0, [P] = 0.0. The concentrations of E, S, ES, and P are graphed using the GRPH command.

4.3 System Structure and Operation

4.3.1 Module descriptions. IBSS is a hierarchical table driven simulation system. The system decomposes into functional subunits or modules. These modules communicate via a common data base composed of the tables used to describe the model and drive the system (Figure 4.8). The modules are the I/O Monitor, Monitor, Equation Compiler, Integrator, Run Record, and Mass Data Handler.

The I/O Monitor interfaces the user to the simulation system. The module receives input from whatever input device is employed and translates it into the monitor commands that will produce the desired response. The module requests information about the model and simulation results through the monitor commands. This module defines the format for user interaction and can vary from a simple teletype monitor to an external device interface with an intelligent terminal. This module can be tailored to the type of peripheral used, and the user's

```
INIT
INPT E + S <->ES  / 1.0 1.0
INPT ES <-> P     / 1.0
INPT E = 1.0
INPT S = 8.0
SDPS 20
RUN 20.0
GRPH 0.0,20,0,E,S,ES,P

CONC                       IBSS GRAPH
0.80D 01 |2                                         444
0.76D 01 |                                        4
0.72D 01 |  2                                  4 4
0.67D 01 |                                 4 4
0.63D 01 |                            4  4
0.59D 01 |     2
0.55D 01 |                          4
0.51D 01 |                        4
0.46D 01 |       2              4
0.42D 01 |                    4
0.38D 01 |                  4
0.34D 01 |          2    4
0.29D 01 |             4
0.25D 01 |          4
0.21D 01 |        4    2
0.17D 01 |      4        2
0.13D 01 |     4            2
0.84D 00 |1      3 3 3 3 3              1 1 1 1 1 1 1 1
0.42D 00 |   4 1 1 1 1 1  1 3 2     1
0.00D 00 | 4                     3 2
         .+---------+---------+-------3-2--333333333333333333
      0.0          0.40D 01  0.80D 01  0.12D 02  0.16D 02  .20D 02

     1-E       2-S        3-ES          4-P           TIME
STOP
```

Figure 4-7 IBSS teletypr Monitor Run

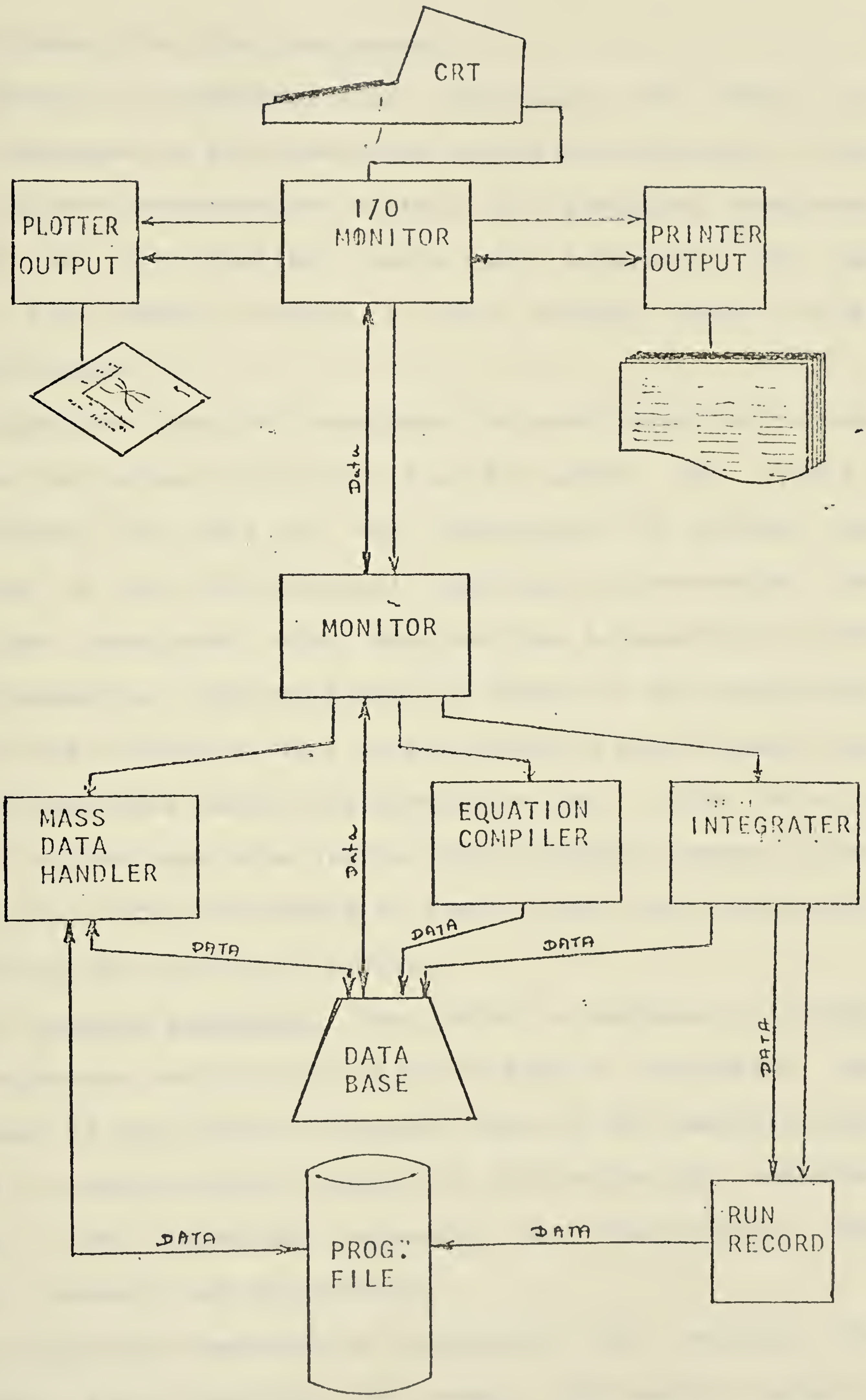


Figure 4-8

IBSS SYSTEM DIAGRAM

choice of formats for that peripheral.

The Monitor is concerned with initiation and control of internal sequences in the simulation system and retrieval of the model's current structure and status. All execution sequences, outside of the I/O Monitor, begin and terminate with the Monitor. The Monitor serves as the control center of the simulation system.

The Equation Compiler translates the model description into entries in the tables in the data base for use by the system. These tables are used by the Integrator to perform the integration of the differential equations representing the model. The Integrator also monitors the integration via the monitor statements. The Run Record is slaved to the Integrator, and stores the simulation data on the system's mass storage area at regular intervals during the simulation run. This data is retrieved by the Mass Data Handler which handles program files, retrieval and storage of models to these files, and retrieval of data saved by the Run Record module.

4.3.2 Control sequences. The system is designed to divorce control structure and data structure as much as possible. The focal point of the control structure lies in the Monitor, which maintains a central control posture by initiating all execution sequences. The execution sequences are Compilation, Run, Retrieval, Command, and Interaction.

The Compilation sequence is concerned with building the data tables that describe the model. The Monitor passes an input statement and control to the Equation Compiler. The statement is then translated into the appropriate table entries.

Control is returned to the monitor with a completion code and any relevant error messages.

The Run sequence performs the actual simulation of the model. The Monitor checks to see that all necessary parameters are set, and transfers control to the Integrator. The Integrator performs a point slope integration of the differential equations representing the model, using the same algorithm employed by David Garfinkel in his simulation system [Garfinkel 1968].

The monitor statements at each integration step are examined to determine whether or not to proceed. At equally spaced intervals during the simulation, determined by the requested number of data points, control is transferred to the run record. The model's state is saved on the system's mass storage area, and control returned to the integrator. This cycle proceeds until the max time limit is reached, a monitor statement returns control to the monitor, or an error occurs. All of these conditions return control to the monitor with a completion code and any relevant error messages.

In the Retrieval sequence, control passes from the Monitor to the Mass Data Handler with a request for retrieval or storage of a model, or simulation data. The mass storage area is searched for the requested information, the data tables updated accordingly, and control returns to the Monitor with a completion code and any relevant error messages.

The Command sequence begins when the Monitor receives a command, and another execution is initiated. If the Monitor receives a request for transfer to a macro command, the command

sequence is initiated with the macro command specified in the request. Macro commands are then executed in sequence. The Compare command can be used for conditional branching in the macro command sequence. When a macro 'Halt' command is executed, or if any error occurs, control is returned to the Monitor.

At the end of a command sequence, the Monitor returns control, with the completion code, any requested information, and any error messages to the requesting module.

The Interaction sequence begins when the monitor passes control with a request for a command to the I/O Monitor. The I/O Monitor presents displays to the user and then passes control to the user. When a reply is supplied by the user, the I/O Monitor formats the reply for the Monitor, and returns control to the Monitor.

4.3.3 Data structure. IBSS is built upon the skeleton provided by the data structures. The data structures are designed to allow both rapid access for numerical methods used during simulation, and dynamic modification of the model for user interaction. The best way to achieve this goal was to design the data structures as set of cross referenced tables. Every effort was made to eliminate redundancy in storage by extensive use of pointers. It is possible to access any information in the system by a path of at most two pointers and an initial index, and in most cases a single pointer and index. Information that is likely to be accessed sequentially, such as the terms for the flux equations, is in an indexed list. Thus the data can be processed rapidly during simulation in a

sequential manner, or indexed during model formulation or modification. Associated with many of the tables are flags which are used to indicate dead entries resulting from model modification during simulation. This reduces the need to compact tables each time a change is made to the structure of the model.

Table 4.9 gives a diagramatic representation of the data structures. Each formula table entry is accessed by the equation name. Each table entry has the equation name, type, and pointers to the rate constants which indicate the position in the equation table for the equation term derived from each half reaction. The terms for each half reaction are given in a list of pointers. Each term consists of pointers to the stoichiometric value and the reactant name. The reactant pointer indicates the position in the reactant table for this reactant.

The reactant table is ordered by reactant name. There is one and only one table entry for each reactant in the system. Each table entry contains the reactant name, present concentration, external input/drain, rate, indicator for a buffered concentration tag, and pointers to the terms for the first derivative of the reactant. There are three pointers for each occurrence of the reactant in a half reaction of the model. The first pointer indicates the position of the term in the equation table giving the positive flux of this reactant. The second pointer indicates the equation table entry for the negative flux. These two pointers also indicate the position of the current value of these terms in the run monitor table. The

REACTANT TABLE ENTRY

d[x]/dt Terms

Reactant Name	Reactant Concentration	External Input	Hold Concentration Constant Tag	▨	+Equation Pointer (Equation Table)	-Equation Pointer (Equation Table)	Stoichiometric Pointer (Data Table)	. . .

EQUATION TABLE ENTRY

flux factors

Rate Evaluation Indicator	▨	Reactant Concentration Pointer (Reactant Table)	. . .

FORMULA TABLE ENTRY

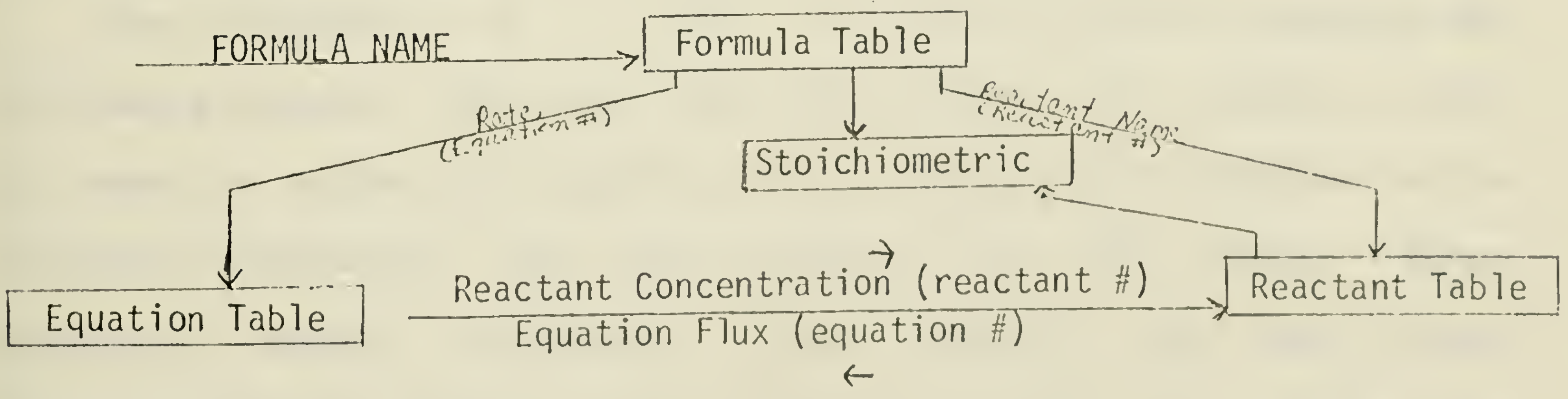
Formula Term

NAME	Rate Pointer to Equation Table	▨	Stoichiometric Pointer (Data Table)	Reactant Pointer (Reactant Table)		Left Half Reaction
TYPE	Rate Pointer to Equation Table	▨	Stoichiometric Pointer (Data Table)	Reactant Pointer (Reactant Table)		Right Half Reaction

DATA TABLE

STOICHIOMETRICS	. . .

ROUTE OF ACCESS [Data accessed (index)]



FORMULA NAME
Formula Table
Rate (Equation #)
Stoichiometric
Reactant Name (Reactant #)
Equation Table
Reactant Concentration (reactant #)
Equation Flux (equation #)
Reactant Table

Figure 4-9 Data structures

third pointer is to the value of the stoichiometric for these terms.

Each entry in the equation table represents a term, less stoichiometric factor, in the reactant's first derivatives. Each half reaction in the formula table produces one entry in the equation table. Each entry has the value for the rate constant, an evaluation indicator set to indicate a 0.0 rate constant, and a list of pointers to the factors for this term. Each factor has two pointers: one indicating a reactant's concentration in the reactant table, and the other the reactant's exponential for this term.

The data table contains lists of values for stoichiometrics and exponential factors. This table is indexed from the formula, equation, and reactant tables.

The data structures are accessed in two different fashions. The first builds, modifies and retrieves information about the model. The data is accessed through the formula table by formula name. The reactant names and concentrations are in the reactant table, the rates and flux terms in the equation table, and reactant stoichiometrics in the data table.

For example, if a term is deleted from an equation the following occurs. The entry for the term in the formula table is used to access the pointers that indicate the entries in the reactant, equation, and data tables for this term. These entries, mostly pointers, are removed and some minor housekeeping is done to the equation table to matain the sructure of this table. The tables are constructed so that there is a minimum amount of shuffling necessary to mantain

their structures.

The second access method calculates the integration values for simulation. The equation table is processed sequentially, yielding the values of the reactant's first derivative's terms. The final value of the first derivative for each reactant is calculated through the first derivative pointers in the reactant table. The calculation for it is made, and the reactants' concentrations are recalculated. This is the basic integration cycle.

Table 4.10 gives a thorough description of the data tables, their structure and interrelation. Table 4.11 shows the compressed format for the data tables when stored in a program file. The data tables are compacted whenever space is needed, or when the program is merged or output.

<u>4.3.4</u> <u>System</u> <u>requirments</u>. The present system requires 90,000 bytes of memory, and runs on an IBM 360/67 under the MTS operating system. This 90,000 bytes of storage divides into 52,000 bytes for the data base, and 38,000 bytes for the program segments (6K for the IO Monitor, 10K for the Monitor, 2K for the integrator, 2K for the Run Record, 8K for the Mass Data Handler, and 10K for the Equation Compiler). These storage requirments may vary depending upon the FORTRAN compiler used to compile the system.

At present the system can handle up to 50 reactants, 50 flux equations with up to 10 terms in each half of the flux equation, up to 30 reaction monitors, and 100 macro statments. These are values of variables to the system, and can be changed when the system is compiled.

Table 4.10. Data Sets

NAME: Name in Data Base
Arrays are given with subscripts

TYPE: I2 = 16 bit integer; L1 = 8 bit Logical (0 or 1); R8 = 64 bit real; I4 = 32 bit integer

Type	Name	Description
	FLDATA	Common name for formula list data set
I2	FORMAX	Max K of FLDATA
I2	FLIMAX	Max I of FLDATA
I2	FSZI (J,K)	I of Formula J,K
I2	STIOF (I,J,K)	Pointer to Stio (DLDATA)
I2	REATF (I,J,K)	Pointer to Reat (RLDATA)
I2	RATEF (J,K)	Pointer to Rate (ELDATA) and associated equations (J=1 Left; J=2 Right)
L1	TYPE (K)	Reaction type (True - reversible; False - forward only)
L1	DTAG (I,J,K)	Denotes "Dead" Reactants
L1	GOODF (K)	Denotes free (Fudge) position in FLDATA
I4	FNAME (K)	Alphanumeric name
	RLDATA	Common name for Reactant List data set
I2	RMAX	Max K for RLDATA
I2	VIMAX	Max I for RLDATA
R8	RL = AT (K)	Reactant names
R8	CONC (K)	Reactant concentrations
I2	VSZ (K)	I for V (1,K) and V (2,K)

I2	VC (K,J,K)	Pointers to contributing equation (ELDATA) J=1, +; J=1 -
I2	STIOR (I,K)	Pointers to Stio (DLDATA)
L1	CONT (K)	Hold concentration constant tag
R8	ADDIW (K)	External reactant input (as rate/time)
I2	GOODR (K)	Denotes # of uses by sytem of reactant GOODR=0 signifies reactant to be deleted
I2	RLIST	Current size of RLDATA (K)
	<u>ELDATA</u>	Common name for equation list data set
I2	ESZI (K)	I for Equation K
I2	REATE (I,K)	Pointer to reactant (RLDATA)
R8	RATE (K)	RATE constant
L1	NOUSE (K)	Signifies 0.0 rate (reverse reaction not in use)
L1	GOODE (K)	Denotes (False) equations to be deleted
I2	ELIST	Current size ELDATA (K)
	<u>DLDATA</u>	Common name for Data List data set
I2	SMAX	Max J of stio (K)
I2	SLIST	Current size of STIO (K)
R8	STIO (K)	Stiometrics
L1	GOODA (K)	Denotes entries to be removed in ALPH
L1	GOODS (K)	Denotes entries to be removed in STIO

	RMDATA	Common name for Run Monitor data set
R8	D1 (K)	dy /dt for reactant y
R8	D1OLD (K)	dy /dt for reactant y
R8	D2 (K)	d^2y /dt for reactant y
I2	CTAG	Compare tag (from DVAL) in CDATA (3)
R8	DT	Current time step dt (CMDATA (2))
R8	TMAX	TIME span for run
R8	MINDT	Minimum dt
R8	ERROR	Max allowable error
R8	TIME	Present system time (CMDATA (1))
I2	COP (J)	Operation: 1-LT, 2-LE, 3-EQ, 4-GE, 5-GT
I2	COPRND (I,J)	Pointer to operands 0 CDATA 0 CONCRM
I2	CVAL (J)	CTAG value
I4	CBRNCH (J)	Branch name
R8	CDATA (L)	Operand data storage. 1=Time, 2=DT, 3=CTAG, 4-10=numeric
I2	CMPSIZ	J
I2	CMPMAX	Max J
I2	CDATSZ	L (initialized to 2)
I2	CDATMX	Max L
I4	CLABEL (J)	Labels for compares
L1	ICMTAG	Internal command mode
I4	BPT	Branch point name
	COMDATA	Common name for communication data set

R8	EROR (I)	Error messages (CHR in first 2 bytes)
I2	LINMAX	Max J
I2	TAG	Numerical tag
L1	EOL	End of line tag
L1	INPUT (J)	Input line
R8	PNAME1	Program name (1 of 2)
R8	PNAME2	Program name (2 of 2)
R8	DPS	No. of data points (plus initial conditions)
	<u>PRGFIL</u>	Common name for internal program file. Rule K=DPS+1; max K of 101
R8	DUMP (I,J,K)	Storage I: 1-Conc, 2-D1, 3-D2, 4-D10LD
	DUMPT (K)	Storage of Time for Dump
	<u>CMDFIL</u>	Command name for internal command data set
I4	CMDNAM (J)	Command names (Labels)
I2	CMD (J)	Commands
L1	CMDOP (I,J)	Command operands (J=1-40)
I2	CMDPLC	Command counter (next command)
I2	CMDSIZ	J
I2	CMDMAX	Max J

Table 4.11. Storage format

MASS DATA STORAGE FORMAT (PROGRAM FILE)

<u>Ranges</u>	<u>Variables</u>
	- File Name (Program Name) -
	SLIST, ALIST, FLIST, RLIST, ELIST, MINDT, ERROR, TMAX, DPS, CMDSIZ, CMPSIZ, CDATSZ

J=1, ELIST
I=1, ESZI (J)
	[ESZI (J), RATE (J), NOUSE (J),
	[REATE (I,J), ALPHE (I,J)]

K=1, RLIST	[REAT (K), VSZ (K), CONT (K), ADDIN (K), GOODR (K), [V (I,1,K), V (I,2,K), STIOR (I,K)]
K-1, FLIST J=1, 2 I-1, FSZI (J,K)	[FNAME (K), TYPE (K), [FSZI (J,K), RATEF (J,K), [STIOF (I,J,K), REATF (I,J,K), ALPHF (I,J,K), DTAG (I,J,K)]

K=1, DPS+1
J=1, RLIST
I-1, 4
	[DUMPT (K),
	[----, *note: this may be moved to first of file if no internal storage available.
	[DUMP (I,J,K)]

J=1, CMPSIZ	[CLABEL (J), COP (J), CVAL (J), COPRND (2,j), COPRND (I,J), CBRNCH (J)]
J=4, CDATSZ	[CDATA (J)]
J=1, CMDSIZ	[CMDNAM (J), CMD (J), [CMD (I,J)]

Any change in these values will result in a change in the size of the data base. A rough estimate of the size of the data base in relation to these variables is

$$SIZE(bytes)=40(T)(F)+R(30+10F)+20MR+50M$$

where T is the maximum number of terms in each half of a flux equation, R is the maximum number of reactants, F is the maximum number of flux equations, MR is the maximum number of reaction monitors, and M is the maximum number of macro statments.

It is possible to segment the program into overlay structures, reducing memory requirments to around 20,000 bytes. It is also possible to segment the data base into three overlays of approximately equal size. The former segmentation could be easily done, while the latter would require modifications to the Mass Data Handler.

The system is designed to be as machine independent as possible. For implementing the system on another machine or under a different operating system, the only modifications necessary would be to the Mass Data Handler, Run Record, and I/O Monitor. These modifications would reflect differences in the I/O routines available under different operating systems, and on different machines.

4.4 Evaluation

IBSS offers the user a comprehensive, easy to use simulation system for the modeling of biochemical systems. It provides simulation within an accuracy range of 1% with reasonable response times. The input format and language provide a descriptive form of modeling biochemical systems. The

system is written in basic FORTRAN, a common language, and is designed to be adaptable to a broad range of hardware configurations.

IBSS is a user oriented system. It can be operated by the naive user with relative ease. The complexities of data structure and recall are handled within the system. IBSS enables the user to recall previous models and analyze their behavior.

IBSS is designed to be used as a test bed for determining the time course behavior of the model under study. IBSS cannot make compensation for faulty data, or erroneous assumptions as to the mechanisms being modeled. But this very fact is useful in checking out the consistency of data and proposed models over a much larger set of circumstances than would be possible in the laboratory.

IBSS might also find use in the simulation of other than biochemical systems. These might include ecological and business systems, where the system can be formulated similarly to reacting chemicals. IBSS could also be used as a teaching aid in the life sciences to demonstrate graphically the interactions and kinetics of biochemical or other biological systems.

The macro command feature of IBSS provides the capability for the model to be structurally self-modifying. Allosteric inhibition can be modeled by changes to the rate constants, or the structure of chemical reactions, as illustrated in the example in this chapter. Chemical equations could be conditionally assembled and incorporated into the system, or

deleted, or modified. This may prove valuable to those wishing to model genetic control mechanisms.

5. AN OPERON MODEL

5.1 Introduction

This chapter will develop a model for genetic regulation based on the lactose operon in *Escherichia coli*. The biological basis for the model will first be described, and then the implementation of the model for IBSS will be explained. The chapter will conclude with a discussion of verification of the model and possible uses for the model.

5.2 Lac Operon

The following material is based on discussions by Hood, Wilson, and Wood [1975], Keeps [1973], and Haggis [1974]. Regulation is a basic function of all organisms and is necessary for the survival of an organism in a changing environment. It is known that *E. coli* growing on glucose medium increases the production of enzymes that metabolise this substrate, while the concentration of inducible enzymes in carbon-source utilization are kept to a minimum. If the bacteria are grown in a lactose medium instead, the concentration of the enzyme beta-galactosidase, which hydrolyzes lactose into galactose and glucose, can increase a thousand-fold. This change in enzyme concentration is referred to as enzyme induction.

In 1961 Jacob and Monod proposed the existence of a new genetic unit, the operon. This is a group of closely related genes on the chromosome that can be controlled in a coordinated way. The operon controlling the enzymes for the utilization of lactose is known as the lac operon. This operon comprises three

structural genes which are transcribed conditionally to produce a single polycistronic messenger RNA molecule.

This mRNA codes for beta-galactosidase, beta-galactoside permease and thiogalactoside transacetylase. The beta-galactosidase, as previously mentioned, hydrolyzes lactose into galactose and glucose, which in turn are used as carbon sources for energy production for the cell. Lac permease is a protein that acts to bring lactose across the cell membrane and into the bacterial cytoplasm. The transacetylase is an enzyme that catalyzes the transfer of one acetyl group from acetyl-CoA to galactose.

The transcription of the three structural genes of the lac operon is controlled by three DNA segments that act as regulatory elements. These are referred to as the lac i gene which codes for lac repressor, the promoter region (p), and the operator (o).

The lac i gene codes for a protein repressor that binds strongly and specifically to to the lac operator region. The repressor also has an active site for the inducer, which in this case is lactose. When the repressor binds with the inducer there is a conformational change which renders the repressor incapable of binding to the operator region of the bacterial DNA.

The promotor is the section of DNA to which the RNA polymerase first becomes attached to initiate the transcription of the structural genes into mRNA. The promotor region is located just before the operator and structural genes on the bacterial DNA. The promotor serves as a regulatory element,

since it controls the rate of mRNA synthesis of a given operon. Both lac structural and i genes have their own promoter. In the model we will refer only to the promoter region for the structural genes.

Since the operator is the section of DNA to which the repressor binds, it must occur in close proximity to the promoter and structural genes of the operon. In the lac operon the operator lies just before the structural genes, and after the promoter region. In this position the RNA polymerase that has attached to the promoter must first pass the operator before transcribing the structural genes. If repressor is bound to the operator, the expression of the lac genes will be repressed. In this fashion the lac operon is regulated by the repressor from the i gene, the promoter and the operator. This is illustrated in Figure 5.1, which is freely adapted from DeRobertis, Saez, and DeRobertis [1975].

The lac operon of _E. coli_ illustrates but one of four possible regulatory schemes using the regulator, promoter, and operator regions on DNA. It is possible to have operons in which the controlling metabolite acts to prevent transcription rather that permit it. These would constitute enzyme repression rather than enzyme induction as in the lac operon. In general, enzyme induction is found in catabolic control, where the enzymes are produced whenever the metabolite becomes avaliable. Enzyme repression is found where the enzymes are used in synthetic pathways. When there is sufficent end product avaliable, transcription of the enzyme genes is curtailed.

In another dimension of operon control, the regulatory

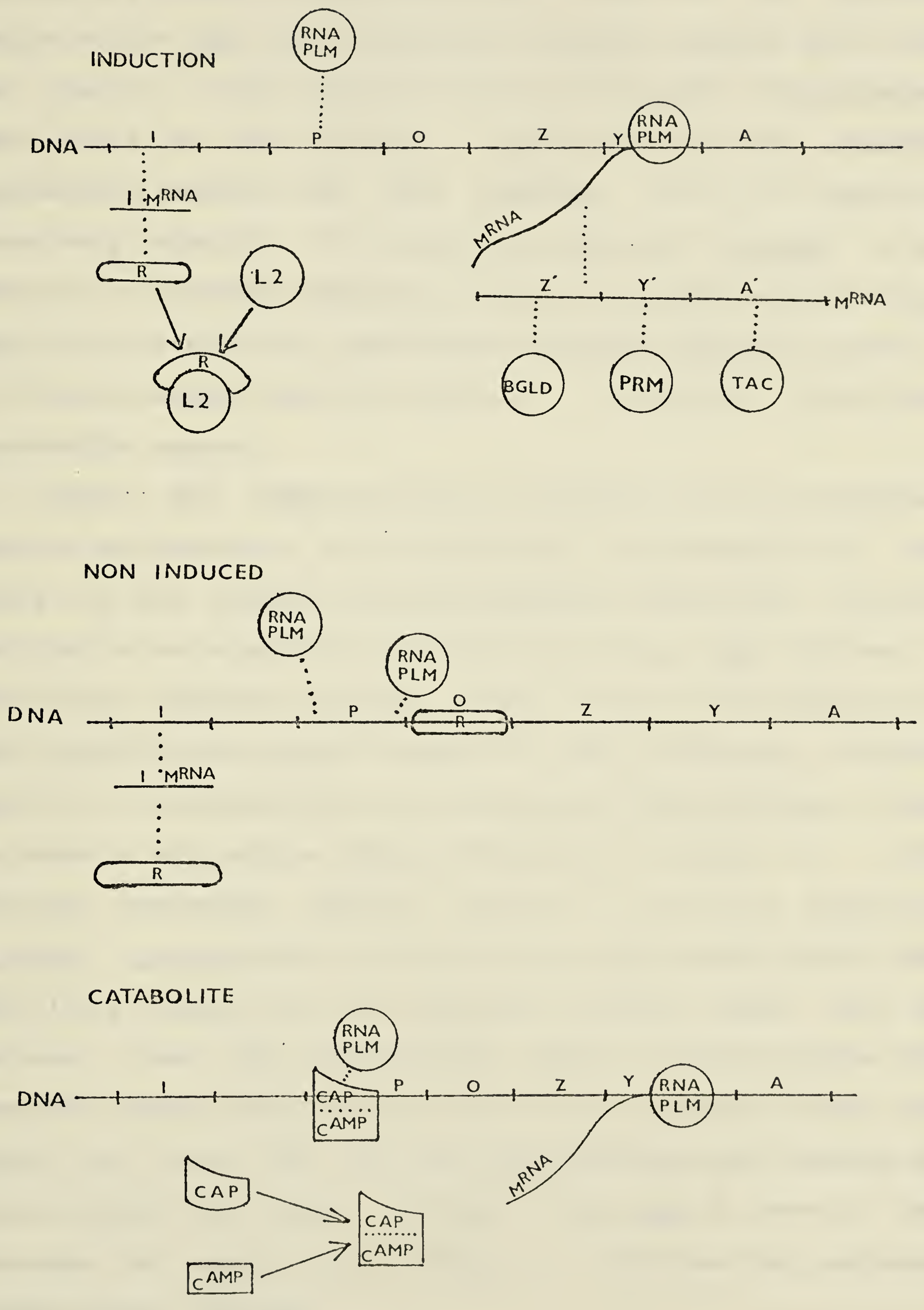


Figure 5-1 Operon and catabolite regulation

protein may prevent transcription when bound to the operator region of the DNA, which constitutes negative control (as in the lac operon) or the regulatory protein may permit transcription when bound to the operator. Three of the four possible regulatory schemes have been observed. The lac operon is negatively inducible. The tryptophan operon in _E. coli_ is an example of negative repression, and the arabinose operon in _E. coli_ is an example of a positively controlled inducible system. To date no example has been found for a positively controlled repressible operon.

Cyclic AMP (cAMP) also has a regulatory role in bacterial operons by activating and deactivating transcription at the level of the promotor in RNA polymerase interaction. This is referred to as catabolite repression and in _E. coli_ involves a catabolite activator protein (CAP), which in the presence of cAMP binds to the promoter region of sugar processing operons enhancing the recognition of this site by RNA polymerase. This is shown in the section labeled CATABOLITE in Figure 5.1. This provides additional positive control of the lac operon in _E. coli_. _E. coli_ grown in glucose has a lower cAMP content than when it is exposed to a less perferred energy source such as lactose. When the intracellular cAMP is low the cyclic AMP receptor protein (CAP) will not bind to the promoter of the lac operon and this will not bind RNA polymerase and therefor no transcription will occur. Therefore if _E. coli_ is grown in the presence of glucose and lactose, it will use the preferred carbon source, glucose.

5.3 The Model

To model the lac operon of _E. coli_, extensive use will be made of the macro features in IBSS. It is possible to model the system solely in terms of flux equations, but this would require a considerable effort both for equation formulation and parameter estimation. The use of the IBSS macro commands eases the task of model formulation and reduces the number of rate constants that need to be determined from laboratory data.

The model was developed by first diagraming the flow of materials, and then drawing a rough flow chart of the control mechanisms. Flux equations were then formulated for material flow. It was then decided not to model translation and transcription directly, but to model these processes by alteration of the flux constants for evolution of the enzymes in the system. Rather than show flux equations for the production of the enzymes, the enzymes fluxes were specified as constant inputs to the system. The control structure was then coded as reaction monitors and macro commands.

In the model that follows, rate constants and some empirical data are omitted. The values for these parameters would have to be obtained from existing literature or from laboratory of _E. coli_ before the simulation of the model could be run. For example, the Cold Spring Harbor Laboratory collection of papers [Beckwith and Zisper, 1973] on the lactose operon contain a wealth of laboratory data on the kinetics of trasncription of the lactose operon, lac mRNA translation, the half life of lac mRNA, and metabolic pathways for lactose metabolism in _E. coli_.

It may be helpful to refer to the Figures 5.2 and 5.3. Figure 5.2 is a flow chart for the control structure of the model. In this figure the macro commands are referenced by the numbers to the upper left of the process boxes. The switch C1 is used to indicate which reaction monitor, C1 or C4, is in use. Figure 5.3 shows the flow of reactants in the model. The numbers on the solid flow lines give the background rates of synthesis of the enzymes. The dashed lines indicate reactant flows not included in the model.

EQUATIONS:

1: L1 + PRM <--> L2 + PRM

2: L2 + BGLD <--> GLC + GLT + BGLD

3: GLT + A + TAC <--> AGLT + TAC

4: GLC + CAMP <--> X

5: PRM --> AMNOA

6: BGLD --> AMNOA

7: TAC --> AMNOA

BUFFERED REACTANTS:

A

Seven flux equations are used. These reflect the transport of lactose (L1) in the medium surrounding the cell by lac permease (PRM) to lactose in the cells cytoplasm (L2) in equation 1. Equation 2 shows the decomposition of the cytoplasmic lactose by beta-galactosidase (BGLD) into galactose (GLT) and glucose (GLC). The transfer of an acetyl group (A) from acetyl-CoA to galactose forming an acetylated galactose (AGLT) by transacetylase (TAC) is shown in equation 3.

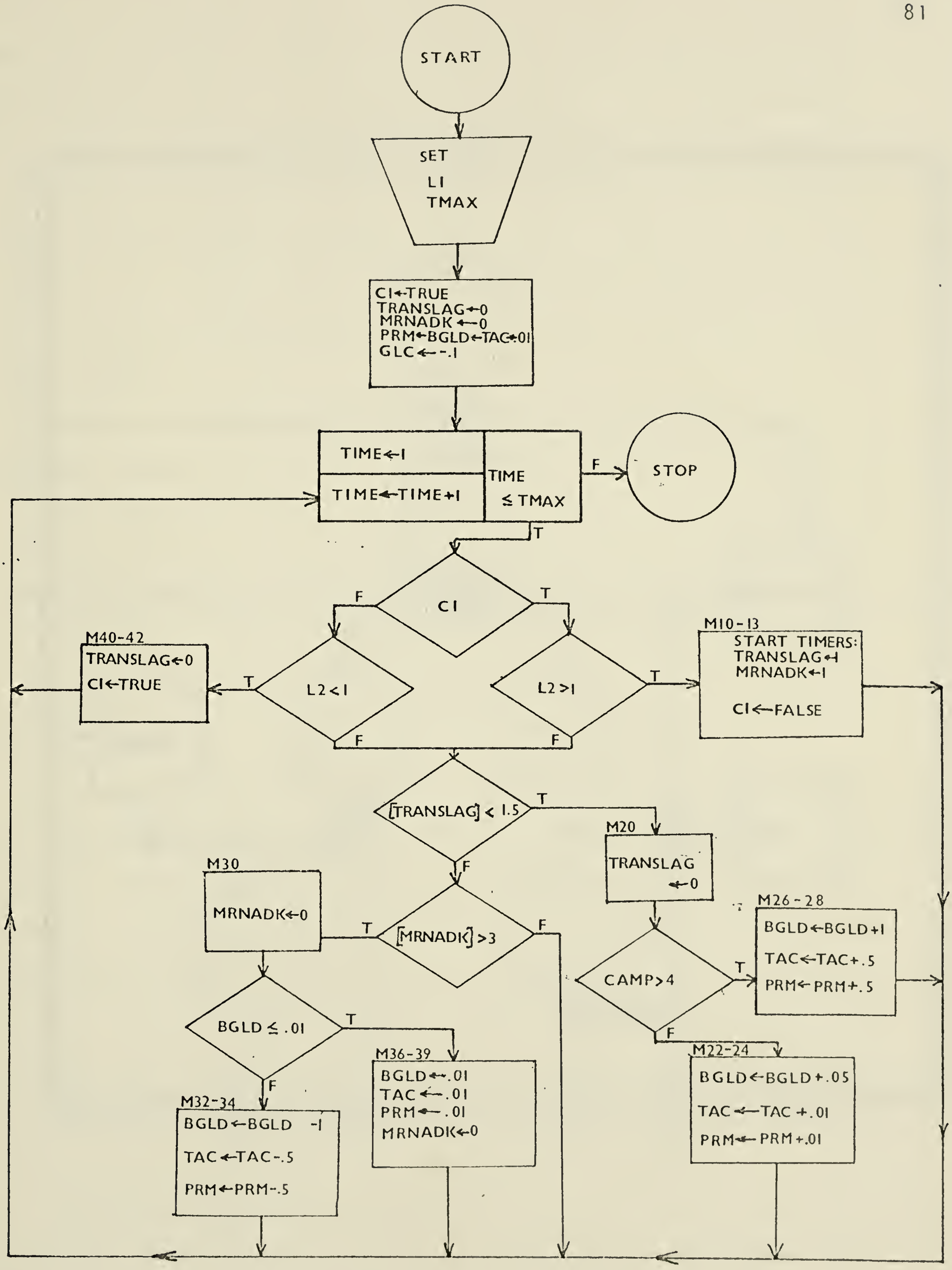


Figure 5-2 Model flowchart

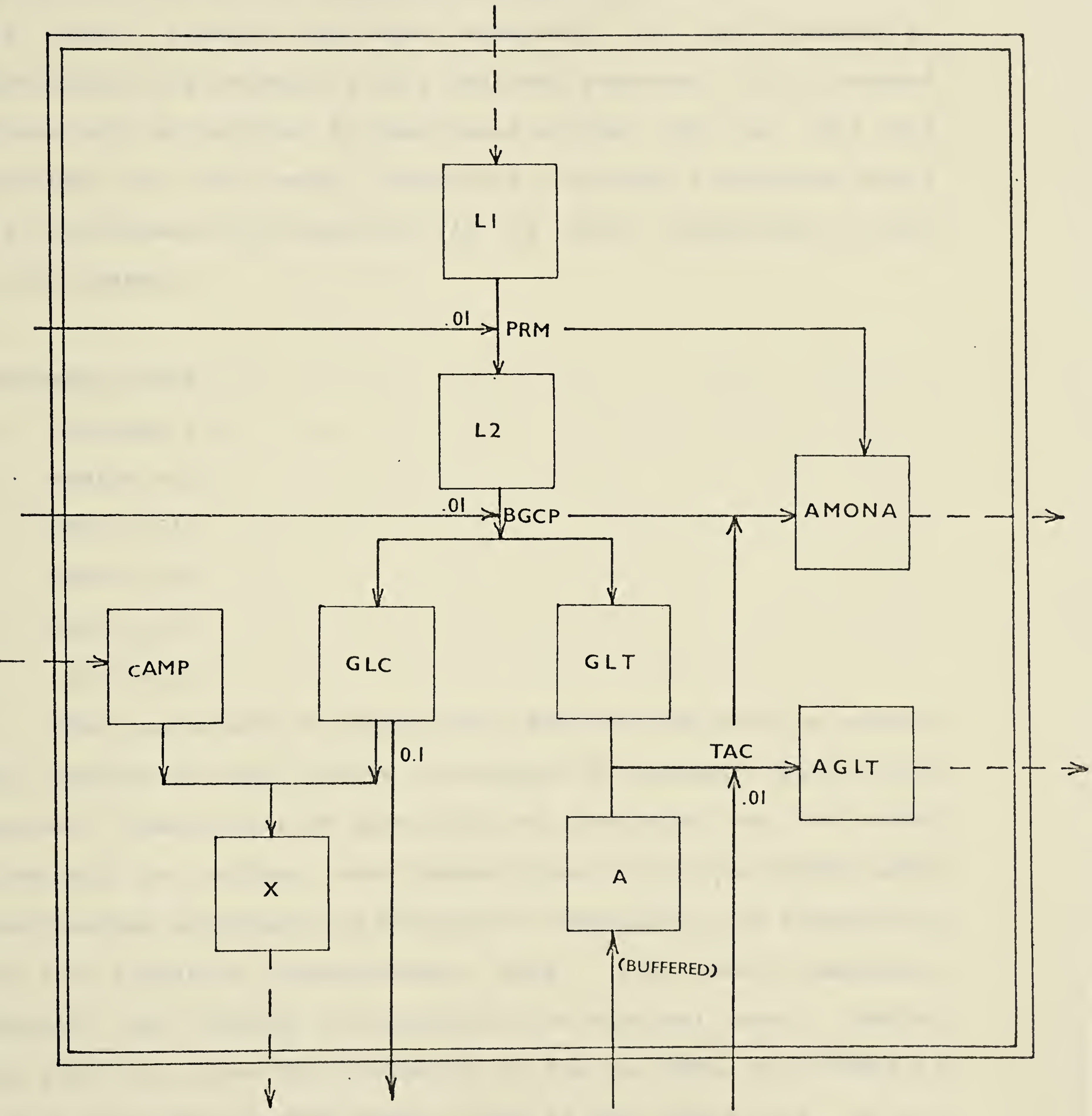


Figure 5-3 Model reactant flow

Equation 4 is used to reflect the reduction of cyclic AMP (CAMP) by glucose for use in catabolite repression. The concentration of acetyl radical (A) from acetyl-CoA is held constant by specifying the reactant A as a buffrerd reactant. It is assumed these will be supplied by functional systems of the cell not included in this model. Equations 5 through 7 represent decay of the enzymes PRM, BGLD and TAC to their constituent amino acids (AMNOA).

```
EXTERNAL INPUT:
    TRANSLAG = 0
    MRNADK = 0
    PRM = .01
    BGLD = .01
    TAC = .01
    GLC = -0.1
```

The production of enzymes PRM, BGLD and TAC will be modeled by regulating their fluxes as constant or external input to the system. These rates of input will be regulated by the macro commands to reflect the transcription of the lac operon under both enzyme induction and catabolite repression, and translation of the resulting polycistronic mRNA. Two other reactants, TRNSLAG and MRNADK, are specified for external input. TRNSLAG is used as a timer for production of the lac mRNA, and MRNADK is similarly used for mRNA decay. When in use, their flux is set to 1, which causes their concentrations to increase in direct proportion with time. This is a bit crude, but at present there are no facilities for variables in the IBSS system other than

reactants. GLUCOSE (GLC) is removed from the system at a constant rate of -.1. This represents utilization of glucose by the cell as an energy source.

REACTION MONITORS:

```
    C1 L2 .GT.  1.0 1,M10
    C2 TRANSLAG .GT.  1.5 1,M20
    C3 MRNADK .GT.  3.0 1,M30
(   C4 L2 .LT.  1.0 1,M40 )
```

Four reaction monitors are used to sense threshold conditions for enzyme regulation. Reaction monitor C1 checks the concentration of cytoplasmic lactose. When the lactose concentration exceeds the threshold, arbitrarily set to 1.0, control is transferred to macro statement M10. Similarly, reaction monitors C2 and C3 cause execution of macro statements M20 and M30 respectively when the given conditions are met. Reaction monitor C4 is bracketed to indicate that it is inserted into the list of reaction monitors by one of the macro statements during model simulation. Reaction monitor C4 is used to sense the drop in concentration of lactose below the arbitrary threshold of 1.0.

MACROS:

```
    M10: SEX       TRANSLAG, 1.0
    M11: SEX       MRNADK, 1.0
    M12: DIMC      C1
    M13: IMC       C4 L2 .LT.  1.0 1, M40
    M14: CONT
```

```
M20: INPUT     TRANSLAG=0
M21: COMP      CAMP .GT.  4.0 1,M25
M22: INC       EXT, BGLD, .05
M23: INC       EXT, TAC, .01
M24: INC       EXT, PRM, .01
M25: CONT
M26: INC       EXT, BGLD, 1.0
M27: INC       EXT, TAC, 0.5
M28: INC       EXT, PRM, 0.5
M29: CONT
M30: INPUT     MRNADK=0
M31: COMP      EX(BGLD) .LE.  .01 1,M36
M32: INC       EXT, BGLD, -1.0
M33: INC       EXT, TAC, -0.5
M34: INC       EXT, PRM, 0.5
M35: CONT
M36: SEX       BGLD, .01
M37: SEX       TAC, .01
M38: SEX       PRM, .01
M39: SEX       MRNADK, 0
M39A: CONT
M40: SEX       TRANSLAG, 0
M41: DIMC      C4
M42: IMC       C1 L2 GT.  1.0 1,M10
M43: CONT
```

The macro statements comprise four subroutines that model enzyme induction and catabolite repression. Macros M10 through M14 will be executed whenever the concentration of cytoplasmic

lactose exceeds the threshold in monitor statement C1. The timers TRANSLAG and MRNADK are turned on by statements M10 and M11. Statement M12 removes monitor statement C1, and statement M13 inserts monitor statement C4. Statement M14 continues the simulation. Statements M20 through M29 are executed whenever sufficient time has passed for transcription of a polycistronic mRNA from the lac operon. Statement M20 resets the TRANSLAG counter to 0. Statement M21 checks the available cyclic AMP (CAMP) concentration. If it is above threshold, statements M26 through M29 are executed, which increases the amount of enzymes BGLAD, TAC and PRM in the system and the simulation continues. If the concentration of CAMP is below threshold, indicating a high concentration of glucose in the system, the amounts of enzymes BGLD, TAC and PRM are increased only slightly by statements M22 through M24, and the simulation is continued by statement M25. This represents catabolite repression.

Macro statements M30 through M39A represent decay of the lac mRNA and subsequently the reduction in the rate of formation of BGLD, TAC and PRM via translation of the lac mRNA. They are executed whenever the MRNADK counter passes threshold value specified in monitor statement C3. Macro statement M30 resets the MRNADK counter. Statement M31 checks to see if the system has stabilized back to a non induced state, and if so, executes statements M36 through M39A. These statements force the rate of formation of enzymes BGLD, TAC and PRM back to backgroung levels (.01) and deactivate the MRNADK counter. Statement M39A continues the simulation. If the system is still in an induced state, statements M32 through M35 are executed. These reduce

the rates of formation of the enzymes BGLD, TAC and PRM to reflect the decay of lac mRNA in the system. Statement M35 continues the simulation.

Macro statements M40 through M43 reset the system after the available lactose has been used up. They are executed when the concentration of cytoplasmic lactose (L2) falls below the threshold established in monitor statement C4. Macro statment M40 turns off the TRANSLAG counter. Statement M41 removes the C4 monitor statement, and statement M42 replaces the lactose threshold monitor statement C1. Statement M43 continues the simulation.

To simulate the model, the previously mentioned statements would be input, and model parameters set by the user. A monitor statement such as

```
C5 CMP TIME .GT.  5 1,M50
```

with macro M50 being

```
M50: HALT
```

would be input. This would let the system run for 5 time steps then return to the user. At this time the lactose concentration (L1) could be set, and monitor C5 changed to

```
C5 CMP TIME .GT.  10 1,M50
```

and the simulation continued.

Existing enzyme PRM would transport the extra-cellular lactose L1 to cytoplasmic lactose L2. Once the threshold concentration of L2 is achieved the production of enzymes BGLD, TAC and PRM will commence via the macro statements M10 through M29.

As long as lactose is available, the enzymes BGLD, TAC and

PRM will continue to be produced. If the concentration of glucose exceeds the threshold established in macro statement M21, the rate of formation of the enzymes will be reduced, reflecting catabolite repression.

Decay of the lac polycistronic mRNA with resulting reductions in the formation of the enzymes will occur via macro statements M30 through M39A and monitor statement C3. After enzyme induction has ceased, the rate of formation of the lac enzymes will return to base levels, and the enzyme concentrations will gradually decrease through enzyme decay.

If the concentration of glucose is initially set high, enzyme induction will be inhibited through the catabolic repression mechanism described by macro statements M21 through M25. If the glucose concentration does drop while lactose is still available, the lac enzymes will be produced, and the model will begin to metabolize lactose.

When time 10 has been passed, the simulation will halt and control will return to the user. The user can then examine the various reactant concentrations and continue or restart the system. Some minor tuning of threshold values will certainly be necessary to produce smooth behavior.

5.4 Verification of the Model and its Uses

In the model proposed in this chapter, no attempt has been made to give exact numeric values for such parameters as rate constants for equations, threshold values for induction or repression, or time delays associatied with mRNA transcription and decay. These values must be obtained from the literature,

or experimental data obtained in the laboratory. Any values specified in the model are for purposes of discussion only.

The whole question of the validity of a model and its simulation results is dependent upon the availability of numerical data by which the model parameters can be estimated and the biological structure determined. If this information is not available, the relationship between the model's simulation results and the actual system being studied is lost.

But if sufficient numerical data is available, simulation can prove a valuable tool. It can allow the researcher to examine the behavior of a model over a wide range of circumstances. While simulation cannot replace experimental observations to prove theories concerning the mechanisms of biological control processes, it can help to guide the researcher through different avenues of investigation.

While the proposed model simulates a negatively induced control system, by changing the monitor statements and macro commands it would be possible to model positively induced, and positively and negatively repressed systems. This would require reversing threshold test, and/or changing the macro statements so that the production of enzymes would normally be on, and turned off only when threshold conditions are met. If the model could be made to agree with experimental observations for the three known regulatory systems, it could then be used to predict the behavior of the positively repressed operon regulation system for which there are no known examples. This could be helpful in directing the search for such a system in the laboratory.

6. CONCLUSIONS AND FUTURE PROSPECTS

Four biochemical simulation systems have been presented. Each makes the task of computer simulation of biochemical models easier. In this sense they all achieve their purpose. How each system handles particular factors of computer simulation distinguishes one system from another. Digital computer systems seem to dominate. The Johnson Foundation's Mark II is the only hybrid discussed and is a one-of-a-kind machine. The basic machine for BIOMOD, Garfinkel's system, and IBSS is a digital computer. BIOMOD requires a fairly large digital computer with rather sophisticated interactive hardware, such as the Rand Tablet and a graphics CRT. Garfinkel's system requires a medium size computer with minimal peripherals. IBSS requires a medium size computer and optimally, a CRT display.

The input/output format for all the systems is adequate to describe the model, but varies somewhat. The patch board method for input in the Mark II is awkward, and the only hard copy is photographs of a precision CRT. Garfinkel's system's formatted card, batch oriented input seems artificial. The tabular and graphical output are very good, but also suffer from the batch orientation of the system. BIOMOD's compartmental approach to model description, coupled with its interactive mode of operation provides a flexible method for input to the system. The quick output of graphical data and availability of hard copy is more than adequate. IBSS uses an interactive mode for input, and takes as input flux equations and macro system commands that allow dynamic modification of the model. Simulation results are

quickly ready as hard or soft copy tabular and graphic output.

The portability of the systems varies significantly. The Johnson Foundation's Mark II is a one-of-a-kind machine, and hybrids with similar capacity are not plentiful. BIOMOD was designed around specialized hardware at RAND, and could require significant reprogramming to be moved to another environment. Garfinkel's system is highly portable, requiring only a medium-sized digital computer and a FORTRAN compiler. IBSS is similar in its portability, being written in FORTRAN. The arrangement of IBSS into small operating sections makes the requirements for memory size relatively low. Overlay structures could be easily employed on smaller computers.

Each system has its advantages and faults. The Mark II is fast, but cumbersome and non-portable. BIOMOD has the most interesting language structure, but also is not portable. Garfinkel's system is portable, but suffers from the batch orientation of the system. IBSS has a good input format and language, is portable and easy to use. The input language is not as comprehensive as BIOMOD's language, but allows description of the model in familiar terms and dynamic monitoring and modification of the model through the macro commands.

IBSS is designed to provide an easy to use tool for biochemical modeling. An interactive format with a clear structure for model description is used to make the system appealing to the novice computer user. The system is written in FORTRAN, so that it can be used on any machine with a FORTRAN compiler. The system is structured into small subunits for clarity and to provide the ability to overlay the system for

smaller computer configurations. Macro command features are provided for those who wish to model control mechanisms in a high-level form. All of these features are designed to make the modeling of biochemical systems on computers a simple task, requiring a minimum of special computer oriented knowledge by the user as has been shown in Chapter five.

It is possible to develop complex models in other simulation systems, but these systems tend to be more general purpose systems. Their input language is not tailored specifically for biochemical description. BIOMOD is the exception, but as prevously mentioned, it is not portable and therefore is not likely to be widely used. To describe models in analog primitives, as in CSMP, or in algorithmic structures, as in FORTRAN or ALGOL, is unacceptable to most biologists.

The macro commands at present are rather primitive. The extension of the macro language to include mathematical expressions and extended logical capabilities would make it much easier to model complex control mechanisms. A suggested syntax for a higher level macro language is of the form:

```
IF X THEN Y ELSE Z
```

where X could be an arbitrarily complicated logical or arithmetic expression, and Y and Z could be expressions that change the model's structure and state. Also to be included would be system variables other than reactant names. This would greately ease the coding of complex mechanisms likely to be encountered in the modeling of large simulations involving an entire organism. This could be achieved either by modifying the monitor to recognize this syntax, or or by inclusion of a separate macro compiler

module to translate the macro syntax into an expanded version of the currently avaliable macros.

Another valuable extension to the system would be a prototype macro facility where a set of macro statements representing a paticular type of control mechanism could be input, and then the paticular reactant names and variables inserted for each use of the prototype in the model. This would eliminate the necessity for coding each use of the control method separately. This also could be implemented by changing the Monitor or inclusion of a pre-processing module for this purpose.

Whatever extensions are made to the system, the effort involved in making them will hopefully be reduced by the modular design of the system. This proved to be very useful during the coding and debugging of the present system.

REFERENCES

Beckwith, J.R., David Zipser, The Lactose Operon, Cold Spring Harbor Laboratory (1970).

Briggs, G.E., and J.B.S Haldane, A note on the Kinetics of Enzyme Action. Biochemical Journal 19:333 (1925).

Chance, B., E.M. Chance, II, and P.H. Seller. Analogue and Digital Representations of Enzyme Kinetics. Journal of Biological Chemistry 235:2440 (1960).

Clark, R.L., and G.F. Groner. The BIOMOD System Implementation. The RAND Corporation, 4-774-NIH (1971).

Clark, R.L., G.F. Groner, and R.A. Berman. The BIOMOD User's Reference Manual. The RAND Corporation, R-746-NIH (1971).

Deland, E.C. Interactive Biochemical Modeling and Analysis, The Rand Corporation, p-4704 (1971).

DeRobertis, E.D.P., and F.A. Saez, E.M.F. DeRobertis, Cell Biology sixth edition, W.B. Saunders Co., Totonto, (1975).

Ellis, T.O., J.F. Heafner, and W.L. Sildey. The GRAIL Project: An Experiment in Man-Machine Communication. The RAND Corporation. RM-599-ARPA, Sept. (1969).

Garfinkel, D. A Machine Independent Language for the Simulation of Complex Chemical and Biochemical Systems. Computers and Biomedical Research2:31 (1968).

Garfinkel, D., J.D. Rutledge, and J.J. Higgens. Simulation and Analysis of Biochemical Systems; I. Representation of Chemical Kinetics. Communications of the Association for Computing Machinery4:559 (1961).

Higgens, J.J. A Special Purpose Analog Computer for Biochemical Research. pp. 101-124 in Computers in Biomedical Research, R.W. Stacy and B.D. Waxman, editors. Academic Press, New York, (1965).

Higgis, G.H. Introduction to Molecular Biology pp. 311-319, Logman Group limited, London (1974).

Hood, Leny E., and John H. Wilson, William B. Wood Molecular Bioloby of the Eucaryotic Cells , W.A. Benjaman Inc., California (1975).

International Business Machines Corporation. System/360, Continuous System Modeling Program (360A-CX-16X) Applications Description. Form No. H20-0240-2, August (1968).

Jacob, F., and j. Monod, On the Regulation of Gene Activity, _Cold Spring Harbor Symposium on Quantitave Boilogy_, 26:193 (1961).

Keeps, A. Sequential Transcription and Translation in the Lactose Operon of Escherichia Coli. pp. 390-406 in _Microbial Genetics_, Morad A. Abou-Sabe editor, Dowden, Hutchson, and Ross Inc., Stroudsburg, Pennsylvania (1973).

Larson, R., P. Sellers, and R. Meyer. Simulation and Analysis of Biochemical Systems: II. Solution of Differential Equations. _Communications of the Association for Computing Machinery_5:63 (1962).

Michaelis, L., and M.L. Menten. Letter. _Biochemical Journal_ 49:333 (1913).

Peterson, L., and D. Garfinkel. Kinetic Simulation Language for Chemistry and Biochemistry. Program # 360D - 03.2008. Program Identification Department of IBM (1968).

APPENDIX A. VERIFICATION OF NUMERICAL ACCURACY

The numerical accuracy was determined by comparisons of the final concentrations of the reactants in the system.

$$E + S \longleftrightarrow ES$$

$$ES \longrightarrow P + E$$

with the initial conditions

$[E] = 1.0$ $[S] = 8.0$

$[ES] = 0.0$ $[P] = 0.0$

when simulated for 20 time steps, with the same data used by Garfinkel. This analysis was considered sufficient, since the same numerical methods are used by both IBSS and Garfinkel's system.

The concentrations agree within 0.08%. The comparison of the concentrations is show in Table A.1. The slight variations can be accounted for by the time at the last step. Garfinkel's system forces the last time step to 20.0000 where IBSS stops the system when the time exceeds or equals 20.0. In this instance that point occured at time 20.00070138 for IBSS.

Table A1. Numerical Accuracy.

REACTANT	IBSS	GARFINKEL	DIFF	% ERROR
E	.9756	.9760	.0004	.02%
S	.04027	.04024	.0003	.08%
ES	.02405	.02404	.00001	.05%
P	7.936	7.936	0.0	0%
			MAX ERROR	.08%

APPENDIX B. MONITOR COMMANDS

A table of the monitor commands is given in Table B1. The command function and action columns give an explanation of each command. The command number is the number by which the monitor recognizes the command. The format is for the operands of the command. The operands appear in the order indicated, with their lengths in bytes indicated above each.

Reply formats for those commands requesting information from the monitor are given in Table B.2. The information occurs in the order indicated, with those items enclosed in brackets indicating repetition of the bracketed items. The length of the entry in bytes is indicated above it.

Table B1. Monitor Commands

<u>ICMON-MONITOR COMMUNICATION FORMATS</u>

<u>Command</u>	<u>Number</u>	<u>Format</u>
Initialize	1	----
Destroy program	2	12/PNAME
Retrieve program	3	12/PNAME, 8/TIME
Merge program	4	12/PNAME
Store program	5	12/PNAME
Set minimum step	6	8/MINDT
RUN	7	8/TMAX
Delete internal input	8	8/REAT
Delete equation	9	4/EQNAME
Delete term	10	4/EQ, 4/SIDE, 4/TERM # side 1=L, 2=R
Add term	11	4/EQ, 4/SIDE, 8/STIO, 8/REAT, 8/ALPH
Set error bounds	12	8/ERROR
Change reactant	13	4/EQ, 4/SIDE, 4/TERM #, 4/---, 8/REACTANT
Change stoichiometric	14	4/EQ, 4/SIDE, 4/TERM #, 4/---, 8/STIO
Change exponential	15	4/EQ, 4/SIDE, 4/TERM #, 4/---, 8/ALPH
Change equation type	16	4/EQ, 4/---, 8/(RATE)
Change rate constant	17	4/EQ, 4/SIDE #, 8/RATE
Delete macro command	18	4/IMCNAME
Model statement	19	248/STATEMENT
Retrieve equation	20	4/EQNAME
Retrieve reactant data	21	8/REACTANT
Retrieve equation names	22	----
Retrieve macro names	23	----

Retrieve reactant names	24	----
Retrieve system state	25	----
Retrieve macro command	26	4/IMCNAME
Set number of sample points	27	4/DPS
DCRT	28	8/REACTANT
Integration Monitor macro	29	4/TAG, 4/---, 4/OP, 8/NAME (OR #), 4/TAG, 8/NAME (OR #), 4/CTAG, 4/BRANCH
HALT	30	----
Continue	31	----
Increment	32	4/TAG, 4/---, 8/NAME, 8/NUMBER
Macro command	33	248/PARAMETERS, 4/LABEL, 4/COMMAND

Table B2. Iomon-Monitor Communication Reply Formats

Command	Number	Format
Retrieve equation	20	4/NAME, 4/# TERMS L, 8/STIO, 8/REAT, 8/ALPH, 4/TYPE, 4/# TERMS R, 8/STIO, 8/REAT, 8/ALPH, 8/FRATE, 8/BRATE Type: 1-Forward; 2-Reversible
Retrieve reactant data	21	8/REAT, 8/CONC, 8/ADDIN, 8/DI, 8/CONT-TAG CONT-TAG: 0-Free; 1-Constant
Retrieve equation	22	4/# EQU, 4/EQUNAME, 4/EQNAME,...
Retrieve macro names	23	4/# IMCS, 4/IMCNAME, 4/IMCNAME,... HALTS #, HALTNAME, HALT NAME,...
Retrieve reactant names	24	4/# REAT, 4/---, 8/REAT, 8/REAT,...
Retrieve system state	25	8/TIME, 4/DPS, 8/MINDT, 8/ERROR,
Retrieve macro command	26	4/NAME, 4/COMMAND #, 40/OPERAND
HALT	30	4/CTAG

APPENDIX C. IBSS COMMANDS

IOMON

I. File Management:

Command	Format
1. Destroy Program	DRPG Prog Name
2. Retrieve Program	RPRG Prog Name, Time, or TIME
3. Merge Program	MPRG Prog Name
4. Store Program	SPRG Prog Name
5. Initialize	INIT

II. Model Description

Command	Format
1. INPUT Equation	INPT Statement
2. Internal Macro Command	IMC Name Statement

III. Execution

Command	Format
1. Set MINDT	SMDT MINDT
2. Set ERROR	SERR ERROR
3. Set NUMBER Of Data Points to Save	SDPS DPS
4. Set External Input	SEX Reactant
5. Run	RUN TMAX
6. Halt	HALT
7. Continue	CONT
8. Compare	COMP NAME. OP. NAME CTAG, BRANCH

IV. Retrieval

	Command	Format
1.	Retrieve Equation Names	REQN
2.	Retrieve Equation	REQU NAME
3.	Retrieve IMC Names	RIMN
4.	Retrieve IMC	RIMC NAME
5.	Retrieve Reactant Names	RRNM
6.	Retrieve Reactant Data (CONC, ADDIN, CONT, D1)	RRDT REACTANT
7.	Retrieve State (TIME, MINDT, ERROR, DPS)	RST

V. Model Manipulation

	Command	Format
1.	Delete Equation	DEQU NAME
2.	Delete External Input	DEIP REACTANT
3.	Delete Constant Reactant Tag	DCRT REACTANT
4.	Delete IMC	CIMC NAME
5.	Increment TAG: 1-Reactant, 2-TMAX, 3-MINDT, 4-FRATE, 5-BRATE, 6-Ext. Input (for 4 and 5 Name is EQUATION NAME)	INCR TAG, NAME, Increment
6.	Delete Term	DIRM EQ, SIDE, TERM #
7.	Add Term	AIRM EQ, SIDE, STIO, REAT, ALPH
8.	Change STIO	CSTO EQ, SIDE, TERM #, STIO
9.	Change Reactant	CRET EQ, SIDE, TERM #, REAT

12